A Study on Cryptographic Protocols: Achieving Strong Security for Zero-knowledge Proofs and Secure Computation

Susumu Kiyoshima

Abstract

This thesis studies *zero-knowledge proofs* and *secure computation*, two of the most fundamental protocols in cryptography. Zero-knowledge proofs are counter-intuitive protocols that allow provers to convince verifiers of the correctness of mathematical statements without revealing any additional knowledge about the statements, and secure computation protocols are powerful protocols that enable mutually distrustful parties to jointly compute any public functions on their secret inputs without compromising the correctness of the outputs and the privacy of the inputs. Zero-knowledge proofs are fundamental in cryptography because they are used as key building blocks in numerous other protocols, and secure computation protocols are fundamental in cryptography because their powerful generality allows us to obtain strong feasibility results about cryptographic protocols.

The focus of this thesis is to obtain new constructions that (provably) satisfy strong security notions such as *concurrent security* and *leakage resilience*. Concurrent security guarantees that a protocol remains secure even when it is executed multiple times in an arbitrary schedule, and leakage resilience guarantees that a protocol remains secure even when adversaries obtain leakages of honest parties' secret internal memories. Concurrent security is motivated by the use of cryptographic protocols on large asynchronous networks like the Internet, and leakage resilience is motivated by the development of various "side-channel" attacks that obtain partial information of honest parties' secret memories via physical measurements on their implementations.

This thesis gives four theoretical results about the problem of achieving strong security at as low cost as possible, where the cost is defined in terms of (asymptotic) efficiency and hardness assumptions. At a high level, these four results can be viewed as a step to solve a fundamental problem about concurrent security and leakage resilience, that is, the problem of constructing secure computation protocols that satisfy concurrent security and leakage resilience with optimal efficiency under minimum assumptions. Specifically, these four results concern natural simplified versions of this fundamental problem (where the simplification is to focus on zero-knowledge protocols rather than secure computation and/or to focus on only either concurrent security or leakage resilience) and show that concurrent security and leakage resilience can be achieved at much lower cost than previously known. Concretely, the results of this thesis are the following.

The first result is about *statistical concurrent non-malleable zero-knowledge arguments*, which are zero-knowledge protocols that currently satisfy the strongest notion of concurrent security in the plain model (i.e., in the model where no trusted third party is available). This thesis shows, in essence, that statistical concurrent non-malleable zero-knowledge protocols can be obtained at no additional cost in terms of hardness assumptions. In other words, this thesis constructs a statistical concurrent non-malleable zero-knowledge argument that is proven secure under the same assumption as the best constructions of (standard) zero-knowledge protocols.

The second result is about *leakage-resilient zero-knowledge arguments*, which are zero-knowledge protocols that satisfy leakage resilience. While the existing constructions are either inefficient in terms of round complexity or secure only under a strong hardness assumption, the construction in this thesis has optimal asymptotic efficiency in terms of round complexity and is secure under a weak hardness assumption.

The third result is about *non-black-box concurrent zero-knowledge arguments*, which are concurrently secure zero-knowledge protocols whose security is proven via a specific technique called *non-black-box simulation*. The motivation behind this result is the long-standing open question of constructing constant-round concurrent zero-knowledge protocols (i.e., concurrent zero-knowledge protocols that have optimal asymptotic efficiency in terms of round complexity), which is known to be solvable only via non-black-box simulation. This thesis gives a construction that has more than constant number of rounds just like the existing construction, but it has an arguably simpler proof of security and therefore can be a useful starting point of future research.

The last result is about *composable secure multi-party computation protocols*, which are secure computation protocols that satisfy concurrent security in a strong sense. The construction in this thesis is proven secure under a well-studied security definition called *angel-based UC security*. Compared with the existing constructions, which are inefficient either because of "non-black-box" use of underlying cryptographic primitives or because of large round complexity, the construction in this thesis is efficient thanks to its "black-box" use of the underlying cryptographic primitives and small round complexity.

Contents

Chapter 1

Introduction

1.1 Zero-knowledge Proofs and Secure Computation

Modern cryptography is not equal to the study of encryption schemes. While classical cryptography only studied the schemes for private communication (namely encryption schemes), modern cryptography also studies the schemes for other various tasks—message authentication, identification, key exchange, and more generally, any tasks that need be done securely over digital networks. Since today's digital life is no longer limited to simple private communication, the study of such schemes is, in modern cryptography, as important as the study of encryption schemes.

The study of *cryptographic protocols*, which constitutes a major part of modern cryptography, involves designing interactive protocols that enable multiple (possibly mutually distrustful) parties to perform predetermined tasks securely. Cryptographic protocols have been studied for various tasks, and some of them have played fundamental roles in modern cryptography. Two main examples of such protocols, which are also those that are studied in this thesis, are *zero-knowledge proofs* and *secure computation*.

Zero-knowledge Proofs. A zero-knowledge (interactive) proof is a somewhat counter-intuitive cryptographic protocol that allows a *prover* to convince a *verifier* of the correctness of a mathematical statement without giving any additional knowledge about the statement. In cryptography, the statement is usually formalized as an instance x of an $\mathcal{NP}$ language L such that the prover knows a witness w for $x \in L$. In this scenario, the prover uses a zero-knowledge proof to convince the verifier that x belongs to L, and the two properties of zero-knowledge proofs, *zero-knowledgeness* and *soundness*, respectively guarantee that no information about w is revealed to the verifier from the proof and that the prover cannot convince the verifier when the statement is false (i.e., when $x \notin L$).[1] The notion of zero-knowledge proofs was introduced by Goldwasser, Micali, and Rackoff in 1985 [GMR85, GMR89] and has been a central object of research in cryptography since then.

[1] Formally, zero-knowledge *proofs* guarantee soundness against any (not necessary polynomial-time) prover, and zero-knowledge *arguments* guarantee soundness only against polynomial-time provers. In this chapter, however, we use the term "zero-knowledge proofs" or "zero-knowledge protocols" to refer to both zero-knowledge proofs and zero-knowledge arguments.

A direct application of zero-knowledge proofs is identification. Suppose that a party, Alice, has a password pwd and another party, Bob, has a hash value $d = h(\mathsf{pwd})$ of pwd, where h is a "secure" hash function (that is, h is a hash function such that finding pwd' such that $h(\mathsf{pwd}') = d$ is hard given d). Now, by using zero-knowledge proofs, Alice can prove her identity to Bob securely as follows: Alice gives a zero-knowledge proof to Bob about the knowledge of pwd such that $h(\mathsf{pwd}) = d$. Bob can be sure about the validity of Alice's identity because of the soundness, and Alice can be sure about the secrecy of pwd against Bob because of the zero-knowledgeness.[2]

A more technical, but equally important, application of zero-knowledge proofs is the use as building blocks in other cryptographic protocols. Very roughly speaking, zero-knowledge proofs are used in other cryptographic protocols to force the protocol participants to follow the protocol specifications. More precisely, in those protocols each party is required to give a zero-knowledge proof about the existence of an input and randomness such that the messages from him/her are correctly computed w.r.t. those input and randomness. (An important point is that this kind of statements can be expressed as an $\mathcal{NP}$ language.) The soundness of the zero-knowledge proof guarantees that each party need to follow the protocol specification to give a convincing proof, so even malicious parties cannot deviate from the protocol specification; on the other hand, the zero-knowledgeness guarantees that the inputs of the honest parties remain hidden even after the proofs are given. Using zero-knowledge proofs in this way to ensure correct behavior is one of the most influential techniques in cryptography, and is a major reason why zero-knowledge proofs are fundamental in the area of cryptographic protocols.

Secure Computation. A secure computation protocol is a powerful protocol that enables mutually distrustful parties to jointly compute any public function f on their secret inputs without compromising the correctness of the outputs and the privacy of the inputs. (In general, f is randomized and reactive, and gives different outputs to different parties.) A classical example of secure computation is Yao's millionaires' problem [Yao82]: Suppose that there is a set of millionaires who wish to know who is the richest but would not like to disclose the exact amounts of their wealth to each other; in this scenario, the millionaires can find the richest person securely by using a secure computation protocol for the following function (assume for simplicity that there is no tie).

$$f(x_1, \ldots, x_N) = i, \text{where } x_i > x_j \text{ for every } j \neq i \ .$$

A more realistic example is conducting statistical analysis on patient data that are stored at multiple hospitals; with secure computation protocols, the hospitals can conduct the analysis without disclosing their patient data to each other.

An advantage of secure computation is its generality, which allows us to obtain strong feasibility results thorough the study of secure computation. Indeed, since any digital task can be formalized as computation of some functions, secure computation

[2] Formally, the underlying zero-knowledge proof is required to have a property called *proof of knowledge* [BG93].

enables us to perform arbitrary task on digital networks, so one can show that any task can be performed securely in a setting by designing a secure computation protocol in that setting. A seminal work by Goldreich, Micali, Wigderson [GMW87] showed that secure computation is possible even in a very severe setting where an arbitrary fraction of the parties are malicious and deviate from the protocol specification.

A research direction about zero-knowledge proofs and secure computation. Even though both zero-knowledge proofs and secure computation have been studied for more than 30 years, they are still actively studied objects in cryptography. A major research direction about them is to give constructions that have stronger security, and this is the topic that we discuss in the next section.

1.2 Quest for Stronger Security

One of the main goals of modern cryptography is to design schemes that have *security proofs* under rigorous security definitions. While classical cryptographic schemes were mainly designed by heuristic and their security often relied on the mere fact that the schemes looked secure, most schemes in modern cryptography are designed by relying on rigorously defined hardness assumptions and have mathematical proofs guaranteeing that the schemes are secure as long as the underlying hardness assumptions hold. The purpose of proving security is to obtain strong confidence in the security—indeed, if a scheme has a security proof under a simple assumption that is easy to study and refute, analyzing the security of that scheme becomes considerably easier. (If the assumption is already well studied, one can also use past experience in the analysis.) A celebrating example of modern cryptography is the work by Goldwasser and Micali [GM84], which showed the first encryption scheme that has a security proof under a rigorous security definition (under an assumption about the hardness of computing quadratic residue modulo composite numbers).

Defining security rigorously is, of course, a very delicate task, and much effort was devoted to obtaining a satisfactory definition for each cryptographic task. Fortunately, cryptography made great success in this process, and by 2000, satisfactory security definitions were obtained for many cryptographic tasks such as encryption schemes [GM84], digital signatures [GMR88], zero-knowledge proofs [GMR85, GMR89], and secure computation [GL91, MR92, Bea92, Can00].

However, the security definitions that were obtained in the early stage of modern cryptography are, while being satisfactory in many settings, not necessarily satisfactory in every setting. Indeed, when cryptographic schemes find new applications, they are often used in a way that is quite different than before, and are sometimes required to have a new security notion that is stronger than the existing ones. An example of this phenomenon is the introduction of *non-malleability* [DDN00]. Originally, encryption schemes were used only for secret communication, and their security notion (such as *semantic security* [GM84]) only concerns the ability to hide information of encrypted messages. If, however, an encryption scheme is used in an online auction where each participant sends a single bid price to the auctioneer in an encrypted form and the one

who bids the highest price wins, the underlying encryption scheme is required to have an additional security guarantee that one cannot transform a ciphertext of a message m into a ciphertext of a related message m'. (If the underlying encryption scheme does not have such a security notion, a participant, Alice, might be able to bit a price that is higher than another participant, Bob, by obtaining Bob's ciphertext via wiretapping and then transforming it to a ciphertext of a higher bit price. Note that Alice does not necessarily break the secrecy of Bob's ciphertext in this attack because Alice does not necessarily learn Bob's bit price during the transformation from Bob's ciphertext.) Non-malleability was introduced as a formulation of this additional security guarantee.

When it turns out that the "standard security definition" of a cryptographic scheme does not provide satisfactory security in a setting, the first task for cryptographers is to strengthen the security definition so that it provides satisfactory security even in that setting. Ideally this strengthening should be done in a way that is as general as possible—that is, in such a way that the strengthened security provides security not only in the specific setting that is considered currently but also in the settings that are obtained by adding conceivable extensions to it. Major examples of such strengthening are the following.

- **Security against stronger computational power.** The first example is perhaps the most natural strengthening: defining security against adversaries that have stronger computational power.

  Statistical security is, informally speaking, security against unbounded-time adversaries. For example, while the basic definition of zero-knowledge proofs [GMR85, GMR89] guarantees that no additional information can be obtained from proofs in polynomial time, *statistical zero-knowledgeness* guarantees that no additional information can be obtained even in unbounded time. (Statistical zero-knowledgeness was also introduced in [GMR89]). An advantage of statistical seucrity is that it is robust against future progress on computation. For example, statistical zero-knowledgeness guarantees that, however faster computers become in future, no additional information will be revealed from proofs.

- **Security under multiple executions.** The second example is defining security under multiple executions of protocols. The motivation behind such security is the use of cryptographic protocols on large asynchronous networks like the Internet: The basic security of most cryptographic protocols, including zero-knowledge proofs and secure computation, is defined in the *stand-alone setting* (i.e., the setting where only a single instance of the protocol is executed at a time) and does not necessarily provide security when multiple instances are executed concurrently; thus, unless there is a guarantee that only a single instance of a protocol is executed at a time on the whole network, the basic security might not provide any security. (Indeed, somewhat counter intuitively zero-knowledge proofs do not necessarily remain zero-knowledge when they are executed concurrently [FS90a].)

  Concurrent security is, informally speaking, security in the setting where protocols are executed multiple times in an arbitrary (possibly maliciously designed) schedule. In the case of zero-knowledge proofs, *concurrent zero-knowledge*

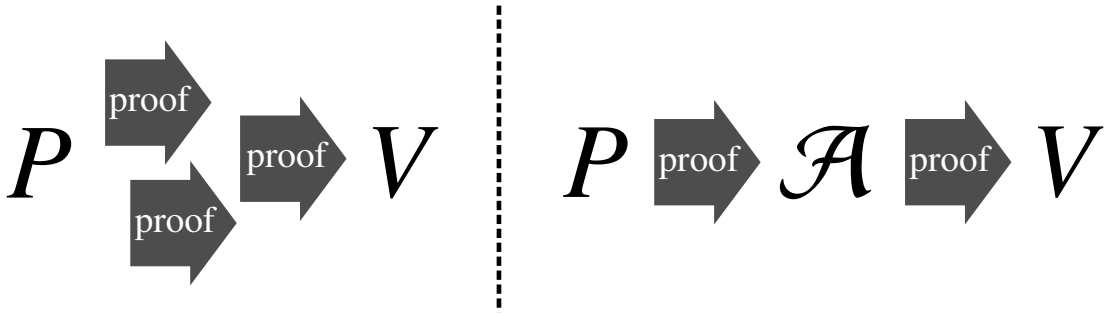

Figure 1.1: Settings for concurrent zero-knowledge (left) and non-malleable zero-knowledge (right). In the figure, P represents the prover, V represents the verifier, and $\mathcal{A}$ represents the man-in-the-middle adversary.

[DNS04] guarantees zero-knowledgeness even when the prover gives multiple proofs to the verifier concurrently in an arbitrary schedule, and *non-malleable zero-knowledge* [DDN00] guarantees zero-knowledgeness even when a *man-in-the-middle adversary* interacts with a prover and a receiver simultaneously (Figure 1.1). In the case of secure computation (and more generally in the case of general cryptographic protocols), more general notions of concurrent security are studied, and the most general one is *universally composable (UC) security* [Can01], which guarantees that a protocol remains secure even when it is concurrently executed with arbitrary (possibly maliciously designed) protocols in an arbitrary schedule.

- **Security with limited randomness resources.** The third example is defining security in the setting where perfect randomness is not always available. The motivation behind such security is the use of cryptographic protocols on computationally weak devices like smart cards, which cannot necessarily produce good randomness by themselves.

  Resettable security is, informally speaking, security in the setting where protocols are executed multiple times with the same randomness. Resettable security was first introduced for zero-knowledge proofs [CGGM00, BGGL01, DGS09] and later extended for secure computation [GS09].

- **Security against "side-channel" attacks.** The last example is defining security in a setting where adversary might employ *side-channel attacks*. Side-channel attacks significantly differ from other attacks in cryptography in that, unlike most cryptographic attacks that obtain information about honest parties through wiretapping or participating in the communication among them, side-channel attacks obtain information about honest parties by measuring physical properties of their devices, such as the running time, power consumption, and electromagnetic radiation [Koc96, KJJ99, QS01]. Surprisingly, side-channel attacks are feasible in practice: For example, a recent work showed that full key recovery of RSA secret keys is feasible by observing sounds on common software and hardware [GST17]. The standard definitions of most cryptographic schemes implicitly assume that

the adversary does not employ any side-channel attack, so it does not guarantee any security against side-channel attacks.

Leakage resilience guarantees security even when some secret information of honest parties is leaked to the adversary. (This leakage to the adversary is introduced to model the information that the adversary obtains via side-channel attacks.) Leakage resilience was first studied for some cryptographic primitives such as encryption schemes and signature schemes, and later extended for zero-knowledge proofs and secure computation (e.g., [GJS11, BGJ+13]).

Once a strengthened notion of security is introduced, the next task for cryptographers is to obtain a construction that satisfies the strengthened security notion at as low cost as possible, where a popular way to define the cost is to define it in terms of efficiency and hardness assumptions. (The ideal goal is of course to achieve the strengthened security notion at the same cost as the basic one.) The main benefit of reducing the cost in terms of efficiency and hardness assumptions is that good efficiency allows us to use the constructions in broad applications, and security under weak hardness assumptions allows us to have strong confidence in the security of the constructions (recall that security proofs tell us that the constructions are secure as long as the underlying hardness assumptions hold).

This research direction—the direction of strengthening security notions and then providing constructions that satisfy the strengthened security notions at as low cost as possible—is one of the main research directions in cryptography, and this thesis follows this research direction with the focus being on concurrent security and leakage resilience.

1.3 Our Results

In this thesis, we show four results about zero-knowledge proof and secure computation with strong security guarantees; the first three are about zero-knowledge proofs and the last one is about secure computation. All of these results are theoretical feasibility results—our focus is to obtain constructions that satisfy the desired security notions with good asymptotic efficiency and under weak hardness assumptions.

At a high level, our results can be viewed as a step to solve a fundamental problem about concurrent security and leakage resilience, that is, the problem of constructing secure computation protocols that satisfy concurrent security and leakage resilience with optimal efficiency under minimum assumptions. (This problem is fundamental because by solving it, we can obtain a strong feasibility result about concurrent security and leakage resilience thorough the generality of secure computation.) Concretely, each of our results concerns a natural simplified version of this fundamental problem, where the simplification is to focus on zero-knowledge protocols rather than secure computation[3] and/or to focus on either concurrent security or leakage resilience (i.e., not to consider them simultaneously).

[3]Studying a security notion on zero-knowledge protocols is a natural first step to study it on secure computation since zero-knowledge protocols are key building blocks of existing secure computation protocols.

In what follows, we give informal descriptions of our results. Formal descriptions of our results can be found in subsequent chapters (see the outline in Section 1.4).

1.3.1 Results about Zero-knowledge Protocols

1.3.1.1 Statistical Concurrent Non-malleable Zero-knowledge from One-way Functions

The first result is about zero-knowledge protocols with very strong concurrent security called *statistical concurrent non-malleable zero-knowledge* (statistical CNMZK).

Statistical CNMZK is a notion that is obtained by combining three security notions that are mentioned in Section 1.2 (namely statistical zero-knowledge, concurrent zero-knowledge, and non-malleable zero-knowledge) and it guarantees that any man-in-the-middle adversary that interacts with multiple provers and verifiers concurrently (Figure 1.2) cannot use any "left proofs" to give meaningful "right proofs," and furthermore the left proofs do not reveal any additional information even in unbounded time. Statistical CNMZK is currently the strongest notion of concurrent security that is achievable for zero-knowledge proofs,[4] and it can be seen as a culmination of a long line of research about statistical zero-knowledge, concurrent zero-knowledge, non-malleable zero-knowledge, and their combinations [BCC88, BCY91, NOVY98, HNO+09, DNS04, RK99, KP01, PRS02, Goy13, CLP13b, PTV14, PPS15, CLP15, DDN00, Bar02, PR08, COSV17, BPS06, GMOS07, LPTV10, LP11a].

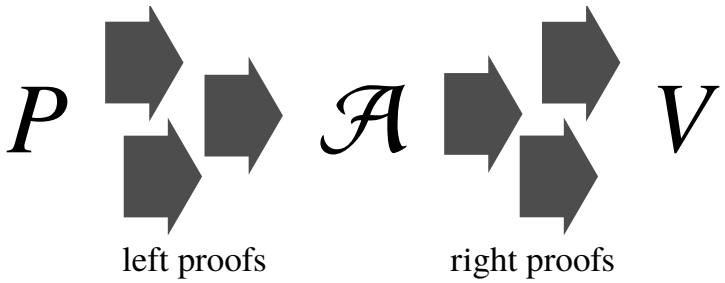

Figure 1.2: Man-in-the-middle adversary for statistical CNMZK. As in Figure 1.1, P represents the prover, V represents the verifier, and $\mathcal{A}$ represents the man-in-the-middle adversary.

The first statistical CNMZK protocol was recently shown by Orlandi et al. [OOR+14], but their protocol has a drawback that it requires a seemingly much stronger hardness assumption than standard stand-alone zero-knowledge protocols. Specifically, even though standard zero-knowledge protocols require only the existence of *one-way functions* (which is the "minimum" assumption in cryptography and known to be necessary for almost all cryptographic schemes such as encryption schemes and signature schemes [IL89, OW93]), the statistical CNMZK protocol of Orlandi et al. requires the *decisional Diffie–Hellman (DDH) assumption*, which is a widely believed assumption in cryptography but seemingly much stronger than the existence of one-way functions.

[4] We restrict our attention to the security notions that are achievable in the *plain model*, where no trusted third party is available.

(A reason why the DDH assumption is considered to be much stronger than the existence of one-way functions is that the DDH assumption implies public-key encryption schemes [ElG85] while it is widely believed that obtaining public-key encryption schemes from one-way functions is very difficult [IR89].) This state-of-the-art immediately raises the following important question: *Does statistical* CNMZK *inherently require stronger assumptions than standard zero-knowledge?*

In this thesis, we show that a statistical CNMZK protocol can be obtained under the same assumption as standard zero-knowledge protocols. That is, we construct a statistical CNMZK protocol under the sole assumption of the existence of one-way functions. The first implication of this result is that the strongest concurrent security of zero-knowledge protocols can be achieved at no additional cost in terms of hardness assumptions. The second implication is that, since the existence of one-way functions is as stated above the minimum assumption in cryptography, the quest for achieving statistical CNMZK protocols from weaker assumptions is essentially completed.

1.3.1.2 Leakage-resilient Zero-knowledge from Collision-resistant Hash Functions

The second result is about leakage-resilient zero-knowledge protocols, which are, as mentioned in Section 1.2, zero-knowledge protocols that are secure against adversaries that obtain secret information of honest parties through physical measurements.

A motivation behind this result is that the existing constructions of leakage-resilient zero-knowledge protocols are not quite satisfactory. The first construction by Garg et al. [GJS11] has a drawback that it has large *round complexity* (that is, the number of interactions between the prover and the receiver is large). Specifically, the construction of Garg et al. has $O(\mathsf{poly}(n))$ rounds when only the existence of one-way functions is assumed, where n is the security parameter[5] and $\mathsf{poly}(\cdot)$ represents a polynomial. The round complexity of their protocol can be reduced to $\omega(\log n)$ if the existence of a little stronger cryptographic primitive, a family of *collision-resistant hash functions*, is assumed, but it is still super-constant. (Another drawback of the construction of Garg et al. is that it only satisfies a slightly relaxed version of leakage resilience.) The subsequent construction by Pandey [Pan14] has an advantage that it has only constant number of rounds, but it has a drawback that it uses a seemingly much stronger assumption than the existence of collision-resistant hash functions (namely the DDH assumption).

A natural question to investigate is whether this unsatisfactory state-of-the-art is inherent—that is, whether leakage-resilient zero-knowledge protocols inherently require either large round complexity or strong hardness assumptions. Given the state-of-the-art, a natural first step to this question is to investigate whether a constant-round leakage-resilient zero-knowledge protocol can be constructed by assuming only the existence of collision-resistant hash functions—in other words, whether it is possible to construct a leakage-resilient zero-knowledge protocol that has the same round complexity as the protocol of Pandey [Pan14] while relying on the same assumption as (the

[5]The security parameter is a parameter that determines the strength of security. Informally speaking, when the security parameter is n, the security holds against all adversaries that run in time polynomial in n.

round-efficient version of) the protocol of Garg et al. [GJS11].

In this thesis, we show that neither large round complexity nor strong hardness assumptions are required for leakage-resilient zero-knowledge protocols. Specifically, we construct a constant-round leakage-resilient zero-knowledge protocol under the assumption of the existence of collision-resistant hash functions. Furthermore we observe that our protocol has an additional advantage that it is *simultaneous leakage-resilient zero-knowledge*, meaning that our protocol satisfies both zero-knowledgeness and soundness in the leakage setting. (The definition of leakage-resilient zero-knowledge only requires that zero-knowledgeness holds in the leakage setting, and indeed the constant-round construction of Pandey [Pan14] is not necessarily sound in the leakage setting.) An implication of this result is that zero-knowledge protocols can have strong security against side-channel attacks even in constant number of rounds and under a very weak assumption.

1.3.1.3 Non-black-box Concurrent Zero-knowledge

The third result is about concurrent zero-knowledge protocols with an additional property that the zero-knowledgeness is proven with *non-black-box simulation*.

This result is motivated by the following central question about concurrent zero-knowledge protocols: *Is it possible to construct a constant-round concurrent zero-knowledge protocol?* Round complexity is, as mentioned above, the number of interactions between the prover and the verifier, and it is one of the most important efficiency factors of cryptographic protocols (this is because communication over networks is often a dominant factor of the running time of protocols). The round complexity of (standard) zero-knowledge protocols is indeed very well studied, and there exist zero-knowledge protocols that have only four rounds under standard assumptions [FS90b, BJY97].[6]

Unlike standard zero-knowledge protocols, the existing constructions of concurrent zero-knowledge protocols require super-constant number of rounds.[7] Concretely, the state-of-the-art is the work by Prabhakaran, Rosen, and Sahai [PRS02], which obtained a $\omega(\log n)$-round construction by refining the analysis of previous works [RK99, KP01]. Although concurrent zero-knowledge protocols have been extensively studied since then, and in particular it has been shown that constant-round concurrent zero-knowledge is possible in various relaxed settings (e.g., the setting where there is an a-priori upper bound on the number of sessions or players [Bar01, GJO+13] and the setting where a relaxed security definition is sufficient [PV08]), the $\omega(\log n)$-round construction of [PRS02] remains to be the best construction for 15 years.

Indeed, there is an evidence that constructing constant-round concurrent zero-knowledge protocols is inherently difficult: There is a negative result stating that a

[6] It is known that two-round zero-knowledge protocols are impossible [GO94], and constructing a three-round zero-knowledge protocol is a major open problem in the area of zero-knowledge proofs. (There exist three-round zero-knowledge protocols that are obtained under strong non-standard assumptions, e.g., [HT98]. In this thesis, we focus on constructions that can be obtained under standard assumptions.)

[7] Again, there exist constant-round concurrent zero-knowledge protocols that are obtained under strong non-standard assumptions [CLP13b, PPS15, CLP15].

constant-round concurrent zero-knowledge protocol is impossible to construct as long as the zero-knowledgeness is proven under a commonly used technical called *black-box simulation* [CKPR02]. Black-box simulation is a technique that is used to prove zero-knowledgeness in the very first work of zero-knowledge proofs [GMR85, GMR89], and had been the only technique to prove zero-knowledgeness for more than a decade. This situation changed dramatically when Barak [Bar01] introduced the first *non-black-box simulation* technique, with which he showed that some of the black-box impossibility results can be overcome. However, constant-round concurrent zero-knowledge protocols have been out of our reach even with his technique (and even with the other non-black-box technique by Bitansky and Paneth [BP12, BP13, BP15]). Indeed, even constructing a super-constant-round concurrent zero-knowledge protocol by using Barak's technique was considered to be very difficult.

A step towards overcoming the black-box impossibility result of constant-round concurrent zero-knowledge was taken by Goyal [Goy13], who showed a non-black-box simulation technique (which is based on that of Barak [Bar01]) that can be used to obtain a concurrent zero-knowledge protocol. His non-black-box simulation technique is quite powerful and was used in a subsequent work to obtain new results about concurrent security [GGS15].

The non-black-box technique of Goyal [Goy13] is however still not powerful enough to be used for constant-round concurrent zero-knowledge protocols. (Indeed, his concurrent zero-knowledge protocol has $\mathsf{poly}(n)$ rounds.) Thus, studying more on non-black-box simulation techniques and developing new ones that can be used for concurrent zero-knowledge is still an important research direction.

In this thesis, we give another non-black-box simulation technique (which is also based on that of Barak [Bar01]) that can be used to obtain a concurrent zero-knowledge protocol. Our technique is, just like the technique of Goyal [Goy13], not powerful enough to be used for constant-round concurrent zero-knowledge protocols. (Indeed, our concurrent zero-knowledge protocol also has $\mathsf{poly}(n)$ rounds.) However, our technique is different from that of Goyal [Goy13], and it is arguably much simpler. Hence, even though our technique itself does not directly lead to constant-round concurrent zero-knowledge protocols, it might become a useful starting point in future.

1.3.2 Result about Secure Computation

The last result is about concurrently secure multi-party computation with an additional advantage that the underlying cryptographic primitives are used only in a "black-box" way. Before elaborating on this result, we first give some backgrounds.

Currently the most powerful notion of concurrent security for secure computation (and for cryptographic protocols in general) is, as mentioned in Section 1.2, UC security [Can01]. UC security guarantees concurrent security in a strong sense since it guarantees security even in the setting where a protocol is executed concurrently with arbitrary (possibly maliciously designed) other protocols in an arbitrary schedule. Furthermore, as observed by [CLP16], UC security also guarantees "environmental friendliness," which guarantees that the security of any other protocol is not adversely affected when they are concurrently executed with any UC-secure protocol.

UC security however has a big drawback that it is too strong to achieve. Specifically, it is known that large classes of functions cannot be computed in the UC-secure way in the standard setting of secure computation, i.e., the setting where no trusted third party is available [CF01, CKL06]. (It is known, however, that computing any function in the UC-secure way is possible if a little help from trusted third party is available [CLOS02].)

Because of this drawback, several alternatives to UC security have been proposed, and one of the most popular ones among them is *angel-based UC security* [PS04, CLP16]. Angel-based UC security is a relaxed version of UC security and considered to guarantee meaningful security in many situations. Angel-based UC security also has an advantage that, unlike UC security, angel-based UC security is possible to achieve without any help from trusted third party [PS04].

The existing constructions of angel-based UC secure computation, however, have disadvantages. Specifically, the constructions by early works [PS04, MMY06, CLP10, CLP16, GLP+15] are not satisfactory since they do not make *black-box use* of the underlying cryptographic primitives, where black-box use of a primitive is the use through the input/output interfaces. (Non-black-box use of a primitive, in contrast, uses the code that implements the primitive, and is considered to be unsatisfactory since it often makes the whole construction computationally inefficient.) The only existing construction that makes only black-box use of the underlying primitives is the one by Lin and Pass [LP12], but their construction is also not satisfactory since it has polynomially many number of rounds.

Summarizing the state-of-the-art, a question that is left open by previous works is whether angel-based UC secure computation inherently requires either non-black-box use of primitives (which leads to poor efficiency) or large round complexity (which also leads to poor efficiency), and this is the question that we answer negatively in this thesis. Concretely, we give a secure multi-party computation protocol that satisfies angel-based UC security, has only $\omega(\log^2 n)$ rounds, and makes only black-box use of the underlying cryptographic primitive.[8] Since black-box use of primitives and small round complexity are both related to efficiency, this result roughly implies that very strong concurrent security of secure computation can be achieved with relatively good efficiency.

1.4 Outline

In Chapter 2, we give basic notions and definitions that are used throughout this thesis. We show our four results formally in Chapters 3, 4, 5, 6, where each of these chapters consists of detailed background, an overview of the techniques, preliminaries, a formal description of the protocol, and a formal proof of security. (Concretely, the result about concurrent non-malleable zero-knowledge is shown in Chapter 3, the result about leakage-resilient zero-knowledge is shown in Chapter 4, the result about non-black-box zero-knowledge is shown in Chapter 5, and the result about secure com-

[8] Just like the construction by Lin and Pass [LP12], our construction can be constructed from a basic cryptographic primitive called *oblivious transfer*.

putation is shown in Chapter 6. These chapters are independent of each other and can be read in any order.) In Chapter 7 we give a conclusion.

Chapter 2

Preliminaries

In this chapter, we give basic notations and definitions that are used throughout this thesis. (The notions and definitions that are used solely in a single chapter are given in the preliminaries section of that chapter.) The description in this chapter is based on those in several textbooks [Gol01, Gol04, KL14].

2.1 Notations

Throughout this thesis, we use n to denote a parameter called *security parameter*. Almost all algorithms in this thesis take the security parameter as input and run in polynomial time in n. (Formally, the security parameter is given as an n-bit unary string 1^n so that the input length is polynomially related with n.)

We use $\mathbb{N}$ to denote the set of all natural numbers, $\mathsf{poly}(\cdot)$ to denote an arbitrary polynomial, and $\perp$ to denote a special error symbol. For any $k \in \mathbb{N}$, we use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. For any randomized algorithm Algo, we use $\mathsf{Algo}(x; r)$ to denote the output of Algo with input x and randomness r, and $\mathsf{Algo}(x)$ to denote the random variable that represents the output of Algo with input x and uniformly chosen randomness. Additionally, for any random variable X, we use $\mathsf{Algo}(X)$ to denote the random variable that represents the output of Algo with input x and uniformly chosen randomness, where x is chosen according to the distribution of X.

We use PPT as an abbreviation of "probabilistic polynomial time," and ITM as an abbreviation of "interactive Turing machine." (Interactive Turing machines are, roughly speaking, Turing machines that have ability to interact with each other via their "communication tapes." For a formal definition, see [Gol01].) For any two ITMs A and B, we use the following notations.

- $\mathsf{trans}\left[A(x) \leftrightarrow B(y)\right]$ is a random variable representing the transcript of the interaction between A and B with input x and y respectively.

- $\mathsf{output}_A\left[A(x) \leftrightarrow B(y)\right]$ (resp., $\mathsf{output}_B\left[A(x) \leftrightarrow B(y)\right]$) is a random variable representing the output of A (resp., B) in the interaction between A and B with input x and y respectively.

- $\mathsf{view}_A\left[A(x) \leftrightarrow B(y)\right]$ (resp., $\mathsf{view}_B\left[A(x) \leftrightarrow B(y)\right]$) is a random variable representing the *view* of A (resp., B) in the interaction between A and B with input x and y respectively, where the view of an ITM during an interaction with another ITM consists of the input and randomness to that ITM plus all the messages that it received from the other ITM during the interaction.

2.2 Basic Definitions

Negligible functions. A function $f(\cdot)$ is *negligible* if f grows slower than the inverse of any polynomial. More precisely, a function f is negligible if for every polynomial $p(\cdot)$, there exists $N \in \mathbb{N}$ such that for every $n > N$, it holds $f(n) < 1/p(n)$. Throughout this thesis, we use $\mathsf{negl}(\cdot)$ to denote an arbitrary negligible function.

Indistinguishability. Two probabilistic ensembles[9] $\mathcal{X} = \{X_k\}_{k\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_k\}_{k\in\mathbb{N}}$ are *computationally indistinguishable*, denoted by $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$, if for every PPT algorithm (or *distinguisher*) D, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $n \in \mathbb{N}$, we have

$$\left|\Pr\left[D(1^n, X_n) = 1\right] - \Pr\left[D(1^n, Y_n) = 1\right]\right| < \mathsf{negl}(n) \ . \tag{2.1}$$

If Equation (2.1) holds for every (computationally unbounded) D, two probabilistic ensembles $\mathcal{X}$ and $\mathcal{Y}$ are *statistically indistinguishable*, denoted by $\mathcal{X} \overset{s}{\approx} \mathcal{Y}$.

One-way functions. A function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following two conditions hold.

- **Easy to compute:** There exists a deterministic polynomial-time algorithm M such that $M(x) = f(x)$ holds for every $x \in \{0,1\}^*$.
- **Hard to invert:** For any PPT algorithm (or *adversary*) $\mathcal{A}$, consider the following probabilistic experiment $\mathrm{Exp}^{\mathrm{inv}}(f, \mathcal{A}, n)$ between $\mathcal{A}$ and a *challenger*.

  1. The challenger chooses uniformly random $x \in \{0,1\}^n$ and computes $y = f(x)$.
  2. On input 1^n and y, the adversary $\mathcal{A}$ outputs x'.

  Then, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $n \in \mathbb{N}$, it holds $\Pr\left[f(x') = y\right] \leq \mathsf{negl}(n)$ in the experiment $\mathrm{Exp}^{\mathrm{inv}}(f, \mathcal{A}, n)$.

[9] A probability ensemble is a family of random variables.

Pseudorandom generators. A deterministic polynomial-time algorithm G is a *pseudorandom generator* if it satisfies the following two properties.

- **Expansion:** There exists a function $\ell : \mathbb{N} \to \mathbb{N}$ such that $\ell(n) > n$ holds for every $n \in \mathbb{N}$, and that $|G(s)| = \ell(n)$ holds for every $s \in \{0,1\}^*$.
- **Pseudorandomness:** For any PPT distinguisher D, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $n \in \mathbb{N}$, we have

$$\left|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]\right| < \mathsf{negl}(n)$$

  where r is chosen uniformly at random from $\{0,1\}^{\ell(n)}$ in the first probability and s is chosen uniformly at random from $\{0,1\}^n$ in the second probability. ◇

Collision-resistant hash functions. A family of functions $\mathcal{H} = \{h_s : \{0,1\}^* \to \{0,1\}^{|s|}\}_{s \in \{0,1\}^*}$ is called *collision resistant* if the following two conditions hold.

- **Easy to compute:** There exists a deterministic polynomial-time algorithm M such that $M(s,x) = h_s(x)$ holds for every $s \in \{0,1\}^*$ and $x \in \{0,1\}^*$.
- **Hard to find collision:** For any PPT adversary $\mathcal{A}$, consider the following probabilistic experiment $\mathrm{Exp}^{\mathrm{coll}}(\mathcal{H}, \mathcal{A}, n)$ between $\mathcal{A}$ and a challenger.

  1. The challenger chooses $h_s \in \mathcal{H}_n$ uniformly at random, where $\mathcal{H}_n \stackrel{\mathrm{def}}{=} \{h_s \in \mathcal{H} \text{ s.t. } |s| = n\}$. (Formally, the challenger chooses $s \in \{0,1\}^n$ uniformly at random.)
  2. On input 1^n and h_s, the adversary $\mathcal{A}$ outputs x, x'.

  Then, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $n \in \mathbb{N}$, it holds $\Pr[x \neq x' \wedge h_s(x) = h_s(x')] \leq \mathsf{negl}(n)$ in the experiment $\mathrm{Exp}^{\mathrm{coll}}(\mathcal{H}, \mathcal{A}, n)$.

We call the functions in a collision-resistant family $\mathcal{H}$ "collision-resistant hash functions."

Remark 2.1. Compared with the definitions in textbooks like [Gol04, KL14], the above definition is simplified since it is assumed that (1) all strings correspond to valid keys (i.e., $h_s \in \mathcal{H}$ exists for every $s \in \{0,1\}^n$) and that (2) the image is $\{0,1\}^n$ when the key length is n (i.e., the image of h_s is $\{0,1\}^n$ when $|s| = n$.) All the results in this thesis hold even when the definitions in [Gol04, KL14] are used.

Decisional Diffie–Hellman (DDH) assumption. Let GenG be a PPT algorithm that, on input 1^n, outputs a description of a cyclic group $\mathbb{G}$, its order q, and a generator $g \in \mathbb{G}$. Then, the DDH assumption on GenG is defined as follows. (In the following, we use $\mathbb{Z}_q$ to denote the set $\{0, \ldots, q-1\}$.)

Definition 2.1 (DDH assumption)**.** ***The DDH assumption holds on*** GenG *if for any* PPT *algorithm* $\mathcal{A}$*, it holds*

$$\left| \Pr\left[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1 \;\middle|\; \begin{array}{l}(\mathbb{G}, q, g) \leftarrow \mathsf{GenG}(1^n);\\ \textit{randomly choose } x, y \in \mathbb{Z}_q\end{array}\right] - \Pr\left[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1 \;\middle|\; \begin{array}{l}(\mathbb{G}, q, g) \leftarrow \mathsf{GenG}(1^n);\\ \textit{randomly choose } x, y, z \in \mathbb{Z}_q\end{array}\right] \right| < \mathsf{negl}(n).$$

We say that ***the DDH assumption holds*** *if there exists a* PPT *algorithm* GenG *such that the DDH assumption holds on* GenG*.* ◇

Remark 2.2. It is well known the DDH assumption implies the existence of one-way functions (this is because the DDH assumption implies the hardness of the discrete logarithm problem). Also, it is well known that the DDH assumption implies the existence of collision-resistant hash function families when the underlying group has a prime order (this is because a family of collision-resistant hash function can be constructed by relying on the hardness of the discrete-logarithm problem in prime-order groups). For details, see textbooks like [KL14].

2.3 Commitment Schemes

In this section, we recall security definitions of *commitment schemes*. In subsequent chapters, we use commitment schemes with various advanced security notions as building blocks of our protocols.

2.3.1 Basic Definitions

We first describe the basic security definitions of commitment schemes. Commitment schemes, often described as a digital equivalent of sealed envelopes, are two-party protocols between a *committer* and a *receiver*. Commitment schemes have two phases: the *commit phase* and the *decommit phase*. In the commit phase, the committer *commits* to a secret input $v \in \{0,1\}^n$ by interacting with the receiver; the transcript of the commit phase is called the *commitment*. In the decommit phase, the committer *decommits* the commitment to v by sending the receiver a message called the *decommitment*; the receiver then outputs either 1 (accept) or 0 (reject). It is required that the receiver accepts the decommitment with probability 1 when both the committer and the receiver behave honestly. Additionally, it is required that the committer cannot decommit a commitment to two different values and that the committed value is hidden from the receiver in the commit phase; the former is called the *binding* property and the latter is called the *hiding* property. Formal definitions of these two properties are given below.

Definition 2.2 (Binding property)**.** *For a commitment scheme* $\langle C, R\rangle$ *and any (not necessarily* PPT*) adversarial committer* C^**, consider the following probabilistic experiment* $\mathit{Exp}^{\mathrm{bind}}(\langle C, R\rangle, C^*, n, z)$ *for any* $n \in \mathbb{N}$ *and* $z \in \{0,1\}^*$*.*

On input 1^n and auxiliary input z, the adversary C^* interacts with an honest receiver in the commit phase of $\langle C, R\rangle$ and then outputs two decommitments, (v_0, d_0) and (v_1, d_1). Then, C^* is said to win the experiment if $v_0 \neq v_1$ but the receiver accepts both (v_0, d_0) and (v_1, d_1) in the decommit phase.

*Then, $\langle C, R\rangle$ is **statistically binding** if for any sequence of auxiliary inputs $\{z_n\}_{n\in\mathbb{N}}$, the probability that C^* wins the experiment $Exp^{\text{bind}}(\langle C, R\rangle, C^*, n, z_n)$ is negligible. If the binding property holds only against PPT adversarial committers, $\langle C, R\rangle$ is said to be **computationally binding**.* ◇

Definition 2.3 (Hiding property)**.** *For a commitment scheme $\langle C, R\rangle$ and any PPT adversarial receiver R^*, consider the following probabilistic experiment $Exp_b^{\text{hide}}(\langle C, R\rangle, R^*, n, z)$ for any $b \in \{0,1\}$, $n \in \mathbb{N}$, and $z \in \{0,1\}^*$.*

On input 1^n and auxiliary input z, the adversary R^* chooses a pair of challenge values $v_0, v_1 \in \{0,1\}^n$ and then interacts with an honest committer in the commit phase of $\langle C, R\rangle$, where the committer commits to v_b. The output of the experiment is the view of R^*

*Let $\mathsf{Exp}_b^{\text{hide}}(\langle C, R\rangle, R^*, n, z)$ denote the output of experiment $Exp_b^{\text{hide}}(\langle C, R\rangle, R^*, n, z)$. Then, $\langle C, R\rangle$ is **computationally hiding** if the following indistinguishability holds.*

$$\left\{\mathsf{Exp}_0^{\text{hide}}(\langle C, R\rangle, R^*, n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{Exp}_1^{\text{hide}}(\langle C, R\rangle, R^*, n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*}$$

*If the indistinguishability holds statistically, $\langle C, R\rangle$ is said to be **statistically hiding**.* ◇

Unless stated otherwise, all the commitment schemes in this thesis are statistically binding and computationally hiding. We say that a commitment is *accepting* if the receiver does not abort in the commit phase, and *valid* if there exists a value to which the commitment can be decommitted (i.e., if there exists a decommitment that the verifier accepts in the decommit phase). A *committed value* of a commitment is a value to which the commitment can be decommitted; we define the committed value of an invalid commitment as $\perp$.

Existing construction: Naor's commitment scheme. In the following we describe Naor's statistically binding commitment scheme, which can be constructed from one-way functions [Nao91, HILL99].

- **Commit phase.** The commit phase consists of two rounds. In the first round, the receiver sends a random $3n$-bit string $r \in \{0,1\}^{3n}$. In the second round, the committer chooses a random seed $s \in \{0,1\}^n$ for a pseudorandom generator $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{3n}$ and then sends $\mathsf{PRG}(s)$ when committing to 0 and sends $\mathsf{PRG}(s) \oplus r$ when committing to 1.

- **Decommit phase.** In the decommit phase, the committer reveals the seed s.

- **Security.** Naor's commitment scheme is statistically binding and computational hiding. Furthermore, the binding and hiding properties hold even when the same first-round message r is used in multiple commitments.

- **Committing to strings.** For any $\ell \in \mathbb{N}$, one can commit to an ℓ-bit string by simply committing to each bit by using Naor's commitment scheme. Furthermore, the binding and hiding properties hold even when the same first-round message r is used in all the commitments.

2.3.2 Extractability

We next recall the definition of *extractable commitment schemes* from [PW09]. Roughly speaking, a commitment scheme is extractable if there exists an expected polynomial-time oracle machine, called *extractor* E, such that for any adversarial committer C^* that gives a commitment to honest receiver, E^{C^*} extracts the committed value of the commitment from C^* as long as the commitment is valid. We note that when the commitment is invalid, E can output an arbitrary garbage value; this is called *over-extraction*. A formal definition is given below.

Definition 2.4 (Extractability)**.** *A commitment scheme $\langle C, R\rangle$ is **extractable** if there exists an expected polynomial-time extractor E such that for any PPT committer C^*, the extractor E^{C^*} outputs a pair (τ, σ) that satisfies the following properties.*

- *τ is identically distributed with the view of C^* that interacts with an honest receiver R in the commit phase of $\langle C, R\rangle$. Let c_τ be the commitment that C^* gives in τ.*

- *If c_τ is accepting, then $\sigma \neq \bot$ except with negligible probability.*

- *If $\sigma \neq \bot$, then it is statistically impossible to decommit c_τ to any value other than σ.* ◇

Existing construction. There exists a four-round extractable commitment scheme based on one-way functions [PW09], which satisfies extractability in a stronger sense: Extractability holds even against adversarial committers that give polynomially many commitments *in parallel*. (The extractor outputs $(\tau, \sigma_1, \sigma_2, \ldots)$ for such committers.) This extractable commitment scheme is described in Figure 2.1.

2.3.3 Concurrent Extractability

We next recall the notion of *concurrently extractable commitment schemes*. Roughly speaking, a commitment scheme is concurrently extractable if there exists a polynomial-time extractor such that for any adversarial committer that commits to polynomially many values concurrently, the extractor can extract the committed values of all the valid commitments from the committer.

Let Com be any two-round statistically binding commitment scheme that can be constructed from one-way functions (e.g., Naor's commitment scheme in Section 2.3.1).

Commit Phase

The committer C and the receiver R take common input 1^n, and C additionally takes private input $v \in \{0,1\}^n$. To commit to v, the committer C does the following with the receiver R.

`commit stage.` C chooses n independent random pairs $\{(a_0^i, a_1^i)\}_{i\in[n]}$ such that $a_0^i \oplus a_1^i = v$ for every $i \in [n]$. Then, C commits to a_0^i and a_1^i for every $i \in [n]$ by using Com. For each $i \in [n]$ and $b \in \{0,1\}$, let c_b^i be the commitment to a_b^i.

`challenge stage.` R sends uniformly random bits $\{e_i\}_{i\in[n]}$ to C.

`reply stage.` C decommits $c_{e_i}^i$ to $a_{e_i}^i$ for every $i \in [n]$.

Decommit Phase

C sends v to R and decommits c_0^i and c_1^i to a_0^i and a_1^i for every $i \in [n]$. Then, R checks whether $a_0^0 \oplus a_1^0 = \cdots = a_0^n \oplus a_1^n = v$.

Figure 2.1: Extractable commitment scheme of [PW09].

Existing construction. There exists a $\omega(\log n)$-round concurrently extractable commitment based on one-way functions [MOSV06]. This is an abstraction of the preamble stage of the concurrent zero-knowledge protocol of Prabhakaran et al. [PRS02], and its extractor performs the extraction by rewinding the adversarial committer according to the carefully designed rewinding strategy of [PRS02, PTV14]. A detailed description of this scheme is given in Figure 2.2. We remark that this scheme has a parameter ℓ, which is the number of extractable commitments that are generated in the commit phase. (In [MOSV06], $\ell = \omega(\log n)$.)

2.3.3.1 Robust Concurrent Extraction Lemma

On the concurrently extractable commitment scheme CECom of Micciancio et al. [MOSV06], a useful lemma called the *robust concurrent extraction lemma* was shown by Goyal et al. [GLP+15]. Roughly speaking, the robust concurrent extraction lemma states that when the adversarial committer additionally participates in an external protocol, the values that are committed to by the adversarial committer can be extracted *without "rewinding" the external protocol*. More precisely, consider any PPT adversarial committer $\mathcal{A}$ that commits to multiple values in concurrent sessions of CECom—these sessions are denoted as the *right sessions*—and simultaneously participates in an execution of an arbitrary protocol $\Pi := \langle B, A\rangle$ with an honest B—this session is de-

CECom is based on the extractable commitment scheme ExtCom of Pass and Wee [PW09] in Figure 2.1, which consists of three stages—`commit`, `challenge`, and `reply`.

Commit Phase

The committer C and the receiver R take common input 1^n and parameter ℓ. (In [MOSV06], $\ell = \omega(\log n)$.) To commit to $v \in \{0,1\}^n$, the committer C commits to v concurrently ℓ times by using ExtCom as follows.

1. C and R execute `commit` stage of ExtCom ℓ times in parallel.
2. C and R do the following for each $j \in [\ell]$ in sequence.
   (a) R sends the `challenge` message of ExtCom for the j-th session.
   (b) C sends the `reply` message of ExtCom for the j-th session.

Decommit Phase

C sends v to R and decommits all the ExtCom commitments.

Figure 2.2: Concurrently extractable commitment of [MOSV06].

noted as the *left session*. The robust concurrent extraction lemma states that for every $\mathcal{A}$, there exists an extractor E that extracts the committed values from $\mathcal{A}$ in every valid right session without "rewinding" the external party B in the left session. The extractor E fails with probability that is exponentially small in $\ell - O(k \log n)$, where ℓ is the parameter of CECom and k is the round complexity of Π. Hence, E fails only with negligible probability if we set $\ell := \omega(k \log n)$.

A formal description of the robust concurrent extraction lemma is given below. (Large parts of the text below are taken from [GLP+15].)

The external protocol Π. Let $\Pi := \langle B, A \rangle$ be an arbitrary two-party protocol. Let $\mathrm{dom}_B(n)$ denote the domain of the input for B and $k := k(n)$ denote the round complexity of Π.

The robust-concurrent attack. Let $x \in \mathrm{dom}_B(n)$. In the *robust-concurrent attack*, the adversary $\mathcal{A}$ interacts with a special (possibly super-polynomial-time) party $\mathcal{E}$ called the *online extractor*. The online extractor $\mathcal{E}$ simultaneously participates in one execution of Π and several executions of CECom, where $\mathcal{E}$ interacts with $\mathcal{A}$ as an honest $B(1^n, x)$ in the execution of Π and interacts with $\mathcal{A}$ as an honest receiver in each execution of CECom. The scheduling of all messages in all sessions—Π as well as CECom—is controlled by $\mathcal{A}$. When $\mathcal{A}$ successfully completes a CECom commitment s, the online extractor $\mathcal{E}$ sends a value α_s to $\mathcal{A}$.

For $n \in \mathbb{N}$, $x \in \mathrm{dom}_B(n)$, $z \in \{0,1\}^*$, let $\mathrm{REAL}^{\mathcal{A}}_{\mathcal{E},\Pi}(n, x, z)$ denote the following probabilistic experiment: On inputs 1^n, x, z, the experiment starts an execution of $\mathcal{A}(1^n, z)$,

which launches the robust-concurrent attack by interacting with $\mathcal{E}(1^n, x, z)$; the output of the experiment is the view of $\mathcal{A}$ and the output of B (who was emulated by $\mathcal{E}$). Let $\mathsf{Real}^{\mathcal{A}}_{\mathcal{E},\Pi}(n, x, z)$ denote the output of $\textsc{Real}^{\mathcal{A}}_{\mathcal{E},\Pi}(n, x, z)$.

The robust concurrent extraction lemma. Roughly speaking, the lemma states that there exists an interactive Turing machine, called the *robust simulator*, that statistically simulates $\mathsf{Real}^{\mathcal{A}}_{\mathcal{E},\Pi}(n, x, z)$ even if the value that the online extractor $\mathcal{E}$ returns to $\mathcal{A}$ at the end of each successful CECom commitment is the committed value of this commitment. Furthermore, the robust simulator does not "rewind" B and runs in time polynomial in the number of the sessions opened by $\mathcal{A}$. A formal statement of the lemma is given below.

Lemma 2.1 (Robust Concurrent Extraction Lemma [GLP+15])**.** *There exists an interactive Turing machine $\mathcal{S}$ called a **robust simulator** such that for every adversary $\mathcal{A}$ and every two-party protocol $\Pi := \langle B, A\rangle$, there exists a party $\mathcal{E}$ called an **online extractor** such that for every $n \in \mathbb{N}$, $x \in dom_B(n)$, and $z \in \{0,1\}^*$, the following conditions hold:*

1. ***Validity constraint.*** *For every view ρ of $\mathcal{A}$ in $\mathsf{Real}^{\mathcal{A}}_{\mathcal{E},\Pi}(n, x, z)$ and for every CECom commitment s appearing in ρ, if there exists a unique value $v \in \{0,1\}^n$ to which the commitment s can be decommitted, then*

$$\alpha_s = v,$$

   where α_s is the value that $\mathcal{E}$ sends to $\mathcal{A}$ at the end of s.

2. ***Statistical simulation.*** *Let $k = k(n)$ be the round complexity of Π. Then the statistical distance between $\mathsf{Real}^{\mathcal{A}}_{\mathcal{E},\Pi}(n, x, z)$ and $\mathsf{output}_{B,\mathcal{S}}\left[B(1^n, x) \leftrightarrow \mathcal{S}^{\mathcal{A}}(1^n, z)\right]$ is given by*

$$\Delta(n) \leq 2^{-\Omega(\ell - k \cdot \log T(n))},$$

   where $\mathsf{output}_{B,\mathcal{S}}\left[B(1^n, x) \leftrightarrow \mathcal{S}^{\mathcal{A}}(1^n, z)\right]$ denotes the joint outputs of $B(1^n, x)$ and $\mathcal{S}(1^n, z)$ after an interaction between them, $\ell := \ell(n)$ is the parameter of CECom, and $T(n)$ is the number of the CECom commitments between $\mathcal{A}$ and $\mathcal{E}$. Furthermore, the running time of $\mathcal{S}$ is $\mathsf{poly}(n) \cdot T(n)^2$.

2.3.4 Non-malleability

We next recall the definition of non-malleable commitment schemes from [LPV08]. For convenience, we use a slightly different presentation (based on indistinguishability rather than simulation), which is used in [LP09, LP11a]. Let $\langle C, R\rangle$ be a tag-based commitment scheme (i.e., $\langle C, R\rangle$ is a commitment scheme that takes an n-bit string—a *tag*—as an additional input). For any man-in-the-middle adversary $\mathcal{M}$, consider the following experiment. On input security parameter $n \in \mathbb{N}$ and auxiliary input $z \in \{0,1\}^*$, $\mathcal{M}$ participates in one left and one right interactions simultaneously. In the left interaction, $\mathcal{M}$ interacts with a committer of $\langle C, R\rangle$ and receives a commitment to value v using identity $\mathsf{id} \in \{0,1\}^n$ of its choice. In the right interaction, $\mathcal{M}$ interacts with a

receiver of $\langle C, R\rangle$ and gives a commitment using identity $\widetilde{\mathsf{id}}$ of its choice. Let $\widetilde{v}$ be the value that $\mathcal{M}$ commits to on the right. If the right commitment is invalid or undefined, $\widetilde{v}$ is defined to be $\perp$. If $\mathsf{id} = \widetilde{\mathsf{id}}$, value $\widetilde{v}$ is also defined to be $\perp$. Let $\mathsf{mim}(\langle C, R\rangle, \mathcal{M}, v, z)$ denote a random variable representing $\widetilde{v}$ and the view of $\mathcal{M}$ in the above experiment.

Definition 2.5 (Non-malleability)**.** *A commitment scheme $\langle C, R\rangle$ is **non-malleable** if for any PPT man-in-the-middle adversary $\mathcal{M}$, the following are computationally indistinguishable.*

- $\{\mathsf{mim}(\langle C, R\rangle, \mathcal{M}, v, z)\}_{n\in\mathbb{N}, v\in\{0,1\}^n, v'\in\{0,1\}^n, z\in\{0,1\}^*}$
- $\{\mathsf{mim}(\langle C, R\rangle, \mathcal{M}, v', z)\}_{n\in\mathbb{N}, v\in\{0,1\}^n, v'\in\{0,1\}^n, z\in\{0,1\}^*}$ ◇

Non-malleability w.r.t. κ-round protocols. We also recall the definition of non-malleability w.r.t. κ-round protocols from [LP09]. In [LP09], this property is also referred to as κ-robustness. We refer to this property as non-malleability w.r.t. κ-round protocols to distinguish it from the κ-robustness for CCA-secure commitment schemes, which is also used in this thesis.

Consider a man-in-the-middle adversary $\mathcal{M}$ that participates in a left interaction—communicating with a machine B—and a right interaction—communicating with a receiver of a commitment scheme $\langle C, R\rangle$, where $\mathcal{M}$ chooses the identity in the right interaction. We denote by $\mathsf{mim}(\langle C, R\rangle, B, \mathcal{M}, y, z)$ the random variable consisting of the view of $\mathcal{M}(z)$ in a man-in-the-middle execution when communicating with $B(y)$ on the left and an honest receiver on the right, combined with the values that $\mathcal{M}(z)$ commits to on the right. Intuitively, we say that $\langle C, R\rangle$ is non-malleable w.r.t. B if $\mathsf{mim}(\langle C, R\rangle, B, \mathcal{M}, y_1, z)$ and $\mathsf{mim}(\langle C, R\rangle, B, \mathcal{M}, y_2, z)$ are indistinguishable whenever interactions with $B(y_1)$ and $B(y_2)$ cannot be distinguished.

Definition 2.6. *Let $\langle C, R\rangle$ be a commitment scheme and B be a PPT ITM. We say that the commitment scheme $\langle C, R\rangle$ is **non-malleable w.r.t.** B if the following holds: For every two sequences $\{y_n^1\}_{n\in\mathbb{N}}$ and $\{y_n^2\}_{n\in\mathbb{N}}$, if for every PPT ITM $\mathcal{A}$ it holds*

$$\left\{\mathsf{view}_{\mathcal{A}}\left[B(1^n, y_n^1) \leftrightarrow \mathcal{A}(1^n, z)\right]\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{view}_{\mathcal{A}}\left[B(1^n, y_n^2) \leftrightarrow \mathcal{A}(1^n, z)\right]\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*},$$

it also holds that for every PPT man-in-the-middle adversary $\mathcal{M}$,

$$\left\{\mathsf{mim}(\langle C, R\rangle, B, \mathcal{M}, y_n^1, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{mim}(\langle C, R\rangle, B, \mathcal{M}, y_n^2, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*}.$$

*We say that $\langle C, R\rangle$ is **non-malleable w.r.t. κ-round protocols** if $\langle C, R\rangle$ is non-malleable w.r.t. any machine B that interacts with the man-in-the-middle adversary in κ rounds.* ◇

2.3.5 CCA Security

We next describe the definitions of *CCA security* and *κ-robustness* of commitment schemes [CLP10, CLP16]. (More precisely, we recall the definitions of CCA security and κ-robustness w.r.t. the committed-value oracle [LP12].)

CCA security. Roughly speaking, a tag-based commitment scheme $\langle C, R\rangle$ (i.e., a commitment scheme that takes an n-bit string—a *tag*—as an additional input) is CCA-secure if it is hiding even against adversary $\mathcal{A}$ that interacts with the following *committed-value oracle*: The committed-value oracle O interacts with $\mathcal{A}$ as an honest receiver in many concurrent sessions of the commit phase of $\langle C, R\rangle$ using tags chosen adaptively by $\mathcal{A}$; at the end of each session, if the commitment of this session is invalid or has multiple committed values, O returns $\perp$ to $\mathcal{A}$; otherwise, O returns the unique committed value to $\mathcal{A}$.

More precisely, CCA-secure commitment schemes are defined as follows. Consider the following probabilistic experiment $\mathrm{IND}_b(\langle C, R\rangle, \mathcal{A}, n, z)$ for each $b \in \{0,1\}$. On input 1^n and auxiliary input z, adversary $\mathcal{A}^O$ adaptively chooses a pair of challenge values $v_0, v_1 \in \{0,1\}^n$ and an n-bit tag $\mathsf{id} \in \{0,1\}^n$. Then, $\mathcal{A}^O$ interacts with the challenger and obtains a commitment to v_b with tag id. Let y be the output of $\mathcal{A}$. The output of the experiment is $\perp$ if $\mathcal{A}$ sends O any commitment using tag id. Otherwise, the output of the experiment is y. Let $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{A}, n, z)$ denote the output of experiment $\mathrm{IND}_b(\langle C, R\rangle, \mathcal{A}, n, z)$.

Definition 2.7 (CCA security)**.** *Let $\langle C, R\rangle$ be a tag-based commitment scheme and O be the committed-value oracle of $\langle C, R\rangle$. Then, $\langle C, R\rangle$ is **CCA-secure (w.r.t the committed-value oracle)** if for any PPT adversary $\mathcal{A}$, the following indistinguishability holds.*

$$\{\mathsf{IND}_0(\langle C, R\rangle, \mathcal{A}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \{\mathsf{IND}_1(\langle C, R\rangle, \mathcal{A}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \quad .$$

$\diamond$

In the experiment, the session between the challenger and $\mathcal{A}$ is called the *left session*, and the sessions between $\mathcal{A}$ and O are called the *right sessions*.

If $\langle C, R\rangle$ is CCA secure only against adversaries that start a single session with O, we say that $\langle C, R\rangle$ is *one-one CCA secure*. That is, one-one CCA security is defined as follows. Let *one-session committed-value oracle* be an oracle that is the same as the committed-value oracle except that it interacts with the adversary only in a single session of the commit phase of $\langle C, R\rangle$. Then, one-one CCA security is defined by replacing the committed-value oracle in the definition of CCA security with the one-session committed-value oracle.

Robustness. Roughly speaking, a tag-based commitment scheme is κ-robust if for any adversary $\mathcal{A}$ and any ITM B, the joint output of a κ-round interaction between $\mathcal{A}^O$ and B can be simulated in polynomial time.

Definition 2.8. *Let $\langle C, R\rangle$ be a tag-based commitment scheme and O be the committed-value oracle of $\langle C, R\rangle$. For any constant $\kappa \in \mathbb{N}$, we say that $\langle C, R\rangle$ is **κ-robust (w.r.t. the committed-value oracle)** if there exists a PPT oracle machine (or **simulator**) $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$ and any κ-round PPT ITM B, the following are computationally indistinguishable:*

- $\left\{\mathsf{output}_{B,\mathcal{A}^O}\left[B(1^n, x, y) \leftrightarrow \mathcal{A}^O(1^n, x, z)\right]\right\}_{n\in\mathbb{N}, x,y,z\in\{0,1\}^n}$

- $\left\{\mathsf{output}_{B,\mathcal{S}^{\mathcal{A}}}\left[B(1^n,x,y)\leftrightarrow\mathcal{S}^{\mathcal{A}}(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$

$\langle C,R\rangle$ *is* ***robust*** *if it is* κ*-robust for any constant* κ*.* $\diamond$

Intuitively, the κ-robustness guarantees that the security of any κ-round protocol (say, the hiding property of a κ-round commitment scheme) holds even against the adversary that interacts with O. In fact, it is easy to see that the following proposition holds.

Proposition 2.1. *Let* $\langle C,R\rangle$ *be a* κ*-robust commitment scheme for a constant* $\kappa\in\mathbb{N}$*, and* B *be any* κ*-round* PPT *ITM. Let* $\{y_n^1\}_{n\in\mathbb{N}}$ *and* $\{y_n^2\}_{n\in\mathbb{N}}$ *be any two sequences such that for every* PPT *adversary* $\mathcal{A}'$*,*

- $\left\{\mathsf{output}_{B,\mathcal{A}'}\left[B(1^n,x,y_n^1)\leftrightarrow\mathcal{A}'(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$ *and*
- $\left\{\mathsf{output}_{B,\mathcal{A}'}\left[B(1^n,x,y_n^2)\leftrightarrow\mathcal{A}'(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$

are computationally indistinguishable. Then, for every PPT *adversary* $\mathcal{A}$*,*

- $\left\{\mathsf{output}_{B,\mathcal{A}^O}\left[B(1^n,x,y_n^1)\leftrightarrow\mathcal{A}^O(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$ *and*
- $\left\{\mathsf{output}_{B,\mathcal{A}^O}\left[B(1^n,x,y_n^2)\leftrightarrow\mathcal{A}^O(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$

are computationally indistinguishable.

Proof. From the definition of κ-robustness, there exists PPT $\mathcal{S}$ such that for each $b\in\{1,2\}$, the following are computationally indistinguishable.

- $\left\{\mathsf{output}_{B,\mathcal{A}^O}\left[B(1^n,x,y_n^b)\leftrightarrow\mathcal{A}^O(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$
- $\left\{\mathsf{output}_{B,\mathcal{S}^{\mathcal{A}}}\left[B(1^n,x,y_n^b)\leftrightarrow\mathcal{S}^{\mathcal{A}}(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$

Also, from the assumption of the proposition, the following are computationally indistinguishable. (Notice that $\mathcal{S}^{\mathcal{A}}$ is PPT since both $\mathcal{A}$ and $\mathcal{S}$ are PPT.)

- $\left\{\mathsf{output}_{B,\mathcal{S}}\left[B(1^n,x,y_n^1)\leftrightarrow\mathcal{S}^{\mathcal{A}}(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$
- $\left\{\mathsf{output}_{B,\mathcal{S}}\left[B(1^n,x,y_n^2)\leftrightarrow\mathcal{S}^{\mathcal{A}}(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$

The proposition follows from these two indistinguishabilities. $\square$

2.4 Interactive Proofs

In this section, we recall security definitions of *interactive proofs* [GMR89]. In subsequent chapters, we use interactive proofs with various advanced security notions as building blocks of our protocols.

2.4.1 Basic Definitions

We first describe the basic security definitions of interactive proof systems. Roughly speaking, an interactive proof system is a two-party protocol between a *prover* and a *verifier* such that the prover can convince the verifier that an instance x belongs to an language L. A formal definition is given below. (In this thesis, we focus on interactive proofs for $\mathcal{NP}$ languages, where the prover is assumed to know a witness w for $x \in L$.)

Definition 2.9 (Interactive Proof System)**.** *For an $\mathcal{NP}$ language L with witness relation $\boldsymbol{R}_L$, a pair of interactive Turing machines $\langle P, V \rangle$ is an **interactive proof** for L if it satisfies the following properties.*

- ***Completeness:*** *For every $x \in L$ and $w \in \boldsymbol{R}_L(x)$,*

$$\Pr\left[\mathsf{output}_V\left[P(x,w) \leftrightarrow V(x)\right] = 1\right] = 1 \quad .$$

- ***Soundness:*** *For every computationally unbounded Turing machine P^*, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $x \notin L$ and $z \in \{0,1\}^*$,*

$$\Pr\left[\mathsf{output}_V\left[P^*(x,z) \leftrightarrow V(x)\right] = 1\right] = \mathsf{negl}(|x|) \quad .$$

*If the soundness condition holds only against every PPT Turing machine, the pair $\langle P, V \rangle$ is an **interactive argument**.* ◇

2.4.2 Witness Indistinguishability

We next describe the definition of *witness indistinguishability* of interactive proofs [FS90a]. Roughly speaking, witness indistinguishability guarantees that a proof with a witness w_1 is indistinguishable from a proof with another witness w_2. A formal definition is given below.

Definition 2.10 (Witness indistinguishability)**.** *An interactive proof (or argument) system $\langle P, V \rangle$ for an $\mathcal{NP}$ language L with witness relation $\boldsymbol{R}_L$ is said to be **witness indistinguishable** if for every PPT adversarial verifier V^* and for every two sequences $\{w_x^1\}_{x\in L}$ and $\{w_x^2\}_{x\in L}$ such that $w_x^1, w_x^2 \in \boldsymbol{R}_L(x)$ for every $x \in L$, the following indistinguishability holds.*

$$\left\{\mathsf{view}_{V^*}\left[P(x,w_x^1) \leftrightarrow V^*(x)\right]\right\}_{x\in L} \overset{c}{\approx} \left\{\mathsf{view}_{V^*}\left[P(x,w_x^2) \leftrightarrow V^*(x)\right]\right\}_{x\in L} \quad .$$

*If the indistinguishability holds statistically, $\langle P, V \rangle$ is said to be **statistically witness indistinguishable**.* ◇

Existing construction: Blum's Hamiltonian-cycle protocol. In the following, we describe (a parallel version of) Blum's Hamiltonian-cycle protocol [Blu86], which is a witness-indistinguishable proof system for the Hamiltonian-cycle problem. To prove that a graph G has a Hamiltonian cycle w, the prover P does the following with the verifier V for n times in parallel, where $n \overset{\text{def}}{=} |G|$.

1. P chooses a random permutation π over the vertices of G, and commits to the adjacency matrix of $\pi(G)$ in a bit-by-bit manner by using any statistically binding commitment scheme.

2. V sends a random bit (or *challenge*) $ch \in \{0,1\}$ to P.

3. **When $ch = 0$:**

   - P sends π to V, and also decommits all the commitments in the first round.
   - V verifies whether the decommitted matrix is equal to the adjacency matrix of $\pi(G)$.

   When $ch = 1$:

   - Among the commitments in the first round, P decommits the ones that correspond to the Hamiltonian cycle w in the adjacency matrix of $\pi(G)$.
   - V verifies whether the decommitted entries of the matrix constitutes a Hamiltonian cycle.

Since statistically binding commitment schemes can be constructed from one-way functions (cf. Section 2.3.1), Blum's protocol can be constructed from one-way functions. When the underlying commitment scheme has k rounds, Blum's protocol has $k + 2$ rounds.

2.4.3 Zero Knowledge

We next describe the definition of *zero-knowledgeness* of interactive proofs [GMR89]. Roughly speaking, zero-knowledgeness guarantees that the verifier cannot learn anything from a proof (except for the fact that x belongs to L), and is formalized by requiring that the view of any (possibly malicious) verifier can be "simulated" by using only the instance x. A formal definition is given below.

Definition 2.11 (Zero-Knowledgeness)**.** *An interactive proof (or argument) $\langle P, V\rangle$ for an $\mathcal{NP}$ language L is **zero-knowledge** if for every PPT adversarial verifier V^*, there exists a PPT algorithm (or **simulator**) $\mathcal{S}$ such that for any sequence $\{w_x\}_{x\in L}$ such that $w_x \in \boldsymbol{R}_L(x)$, the following indistinguishability holds.*

$$\{\mathsf{view}_{V^*}\,[P(x,w_x) \leftrightarrow V^*(x,z)]\}_{x\in L, z\in\{0,1\}^*} \overset{c}{\approx} \{\mathcal{S}(x,z)\}_{x\in L, z\in\{0,1\}^*} \quad .$$

◇

2.4.4 Proof of Knowledge

We next describe the definition of *interactive proofs of knowledge* [BG93]. Roughly speaking, an interactive proof of knowledge system is an interactive proof system such that the prover can convince the verifier, not only that x belongs to L, but also that the prover has a witness for $x \in L$. A formal definition is given below.

Definition 2.12 (Proof of Knowledge)**.** *An interactive proof system $\langle P, V\rangle$ for an $\mathcal{NP}$ language L with witness relation $\boldsymbol{R}_L$ is said to be **proof of knowledge** if there exists an expected PPT oracle machine (or **extractor**) E such that the following holds: For every computationally unbounded Turing machine P^*, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $x \in \{0,1\}^*$ and $z \in \{0,1\}^*$,*

$$\Pr\left[\exists w \in \boldsymbol{R}_L(x) \text{ s.t. } E^{P^*(x,z)}(x) = w\right] > \Pr\left[\mathsf{output}_V\left[P^*(x,z) \leftrightarrow V(x)\right] = 1\right] - \mathsf{negl}(|x|) \ .$$

*If the above condition holds only against every PPT Turing machine P^*, the interactive proof system $\langle P, V\rangle$ is said to be **argument of knowledge**.* ◇

2.5 Universal Arguments

In this section, we recall the definition of *universal argument* systems [BG08], which we use as building blocks of our protocols.

Universal language. For the purpose of this thesis, it suffices to give the definition of universal arguments only w.r.t. the membership of a single "universal" language $L_{\mathcal{U}}$. For triplet $y = (M, x, t)$, we have $y \in L_{\mathcal{U}}$ if non-deterministic machine M accepts x within t steps. (Here, all components of y, including t, are encoded in binary.) Let $\mathbf{R}_{\mathcal{U}}$ be the witness relation of $L_{\mathcal{U}}$, i.e., $\mathbf{R}_{\mathcal{U}}$ is a polynomial-time decidable relation such that for any $y = (M, x, t)$, we have $y \in L_{\mathcal{U}}$ if and only if there exists $w \in \{0,1\}^{\leq t}$ such that $(y, w) \in \mathbf{R}_{\mathcal{U}}$.

Universal argument. Roughly speaking, universal arguments are "efficient" arguments of knowledge for proving the membership in $L_{\mathcal{U}}$, where they are efficient in the sense that the prover's running time is bounded by the time that is needed for verifying the validity of the witness that the prover has. A formal definition is given below. In the following, for any $y = (M, x, t) \in L_{\mathcal{U}}$, we use $T_M(x, w)$ to denote the running time of M on input x with witness w, and let $\mathbf{R}_{\mathcal{U}}(y) \stackrel{\text{def}}{=} \{w : (y, w) \in \mathbf{R}_{\mathcal{U}}\}$.

Definition 2.13 (Universal argument)**.** *A pair of interactive Turing machines $\langle P, V\rangle$ is a **universal argument** system if it satisfies the following properties.*

- ***Efficient verification:*** *There exists a polynomial p such that for any $y = (M, x, t)$, the total time spent by (probabilistic) verifier strategy V on inputs y is at most $p(|y|)$.*
- ***Completeness by a relatively efficient prover:*** *For every $y = (M, x, t) \in L_{\mathcal{U}}$ and $w \in \boldsymbol{R}_{\mathcal{U}}(y)$,*

$$\Pr\left[\mathsf{output}_V\left[P(y,w) \leftrightarrow V(y)\right] = 1\right] = 1 \ .$$

  Furthermore, there exists a polynomial q such that the total time spent by P, on input (y, w), is at most $q(|y| + T_M(x, w)) \leq q(|y| + t)$.

- ***Computational Soundness:*** *For every* PPT *Turing machine P^*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every $y = (M, x, t) \notin L_{\mathcal{U}}$ and $z \in \{0,1\}^*$,*

$$\Pr\left[\mathsf{output}_V\left[P^*(y,z) \leftrightarrow V(y)\right] = 1\right] < \mathsf{negl}(|y|) \ .$$

- ***Weak Proof of Knowledge:*** *For every polynomial $p(\cdot)$ there exists a polynomial $p'(\cdot)$ and a* PPT *oracle machine E such that the following holds: For every* PPT *Turing machine P^*, every sufficiently long $y = (M, x, t) \in \{0,1\}^*$, and every $z \in \{0,1\}^*$, if $\Pr\left[\mathsf{output}_V\left[P^*(y,z) \leftrightarrow V(y)\right] = 1\right] > 1/p(|y|)$, then*

$$\Pr_r\left[\exists w = w_1 \cdots w_t \in \boldsymbol{R}_{\mathcal{U}}(y) \text{ s.t. } \forall i \in [t], E_r^{P^*(y,z)}(y,i) = w_i\right] > \frac{1}{p'(|y|)} \ ,$$

  where $E_r^{P^*(y,z)}(\cdot,\cdot)$ denotes the function defined by fixing the randomness of E to r, and providing the resulting E_r with oracle access to $P^*(y,z)$. ◇

The weak proof-of-knowledge property of universal arguments only guarantees that each individual bit w_i of a witness w can be extracted in probabilistic polynomial time. However, for any $y = (M, x, t) \in L_{\mathcal{U}}$, since any witness $w \in \mathbf{R}_{\mathcal{U}}(y)$ is of length at most t, there exists an extractor (called the *global extractor*) that extracts the whole witness in time polynomial in $\mathsf{poly}(|y|) \cdot t$. We call this property the *global proof-of-knowledge property* of a universal argument.

Chapter 3

Statistical Concurrent Non-malleable Zero-knowledge from One-way Functions

In this chapter, we show our first result: A statistical concurrent non-malleable zero-knowledge argument based on one-way functions.

3.1 Background

As one can see in Definition 2.11, the zero-knowledge property of interactive proofs/arguments is defined in the stand-alone setting, so the adversarial verifier is assumed to interact with a single prover at a time.

Non-malleable zero-knowledge (NMZK) [DDN00] and *concurrent zero-knowledge* (CZK) [DNS04] are two well-known notions of the ZK property in the concurrent setting. In the setting of NMZK, the adversary concurrently interacts with an honest prover in the *left session* and an honest verifier in the *right session*, and in the setting of CZK, the adversary concurrently interacts with unbounded number of honest provers.

As a security notion that implies both NMZK and CZK, Barak et al. [BPS06] proposed *concurrent non-malleable zero-knowledge* (CNMZK). CNMZK guarantees the ZK property in the setting where the adversary concurrently interacts with multiple provers in the left sessions and multiple verifiers in the right sessions. More precisely, it guarantees that receiving proofs in the left session does not "help" the adversary to give proofs in the right sessions—that is, if the adversary can prove some statements in the right sessions while receiving proofs in the left sessions, the adversary could prove the same statements even without receiving proofs in the left sessions. In the definition of CNMZK, this guarantee is formalized as the existence of a *simulator-extractor* that can simulate the adversary's view in the left and right sessions while extracting witnesses from the adversary in the simulated right sessions.

The first CNMZK argument for $\mathcal{NP}$ was constructed by Barak et al. [BPS06]. Subsequently, a computationally efficient construction was shown by Ostrovsky et al. [OPV10]. The first CNMZK *proof* was constructed by Lin et al. [LPTV10], and a variant of their protocol was shown to be secure with adaptively chosen inputs by Lin and

Pass [LP11a]. Additionally, a CNMZK argument that is secure with "fully" adaptively chosen inputs was recently constructed by Venkitasubramaniam [Ven14].

Very recently, Orlandi et al. [OOR+14] constructed the first *statistical* CNMZK argument, i.e., a CNMZK argument such that the simulator-extractor outputs view that is statistically indistinguishable from the adversary's real view. Statistical CNMZK is clearly of great interest since it guarantees quite strong security in the concurrent setting. However, statistical CNMZK is hard to achieve, and the existing techniques of computational CNMZK protocols seem to be insufficient for constructing statistical CNMZK protocols (see Section 3.2.1).

An important question on statistical CNMZK protocols is what hardness assumption is needed for constructing them. The statistical CNMZK argument of Orlandi et al. [OOR+14] was constructed under the DDH assumption (or the existence of dense cryptosystems). Hence, we already know that statistical CNMZK protocols can be constructed under standard assumptions. However, since the existence of one-way functions is sufficient for constructing both statistical ZK protocols and computational CNMZK protocols [HNO+09, BPS06], it is natural to ask the following question.

> *Can we construct statistical concurrent non-malleable zero-knowledge protocols by assuming only the existence of one-way functions?*

3.1.1 Our Result

In this chapter, we answer the above question affirmatively.

Theorem 3.1. *Assume the existence of one-way functions. Then, there exists a statistical concurrent non-malleable zero-knowledge argument for $\mathcal{NP}$ with round complexity* $\mathsf{poly}(n)$. *Furthermore, if there exists a family of collision-resistant hash functions, the round complexity can be reduced to* $\omega(\log n)$.

The round complexity of our statistical CNMZK argument—$\mathsf{poly}(n)$ rounds when only the existence of one-way functions is assumed and $\omega(\log n)$ rounds when the existence of a family of collision-resistant hash functions is assumed—is the same as the round complexity of the known statistical CZK arguments [GMOS07]. Thus, our result closes the gap between statistical CNMZK arguments and statistical CZK arguments. Furthermore, since the security of our statistical CNMZK protocol is proven via black-box simulation, the logarithmic round complexity of our hash-function-based protocol is essentially tight due to the lower bound on black-box CZK protocols [CKPR02].

3.1.2 Outline

In Section 3.2, we give an overview of our techniques. In Section 3.3, we give the notations and definitions that are used specifically in this chapter. In Section 3.4, we describe our statistical CNMZK argument and prove its security. In Section 3.5, we give supplementary materials about this chapter.

3.2 Overview of Our Techniques

In this section, we give an overview of our techniques.

3.2.1 Previous Techniques

We start by describing the difficulty of constructing statistical CNMZK protocols using the techniques of existing computational CNMZK protocols [BPS06, LPTV10].

First, let us recall the protocols of [BPS06, LPTV10]. The definition of CNMZK requires the existence of a simulator-extractor that can simulate the adversary's view while extracting witnesses for the statements proven by the adversary. To satisfy this definition, CNMZK protocols need to satisfy the following properties: (i) the proofs in the left sessions can be simulated for the adversary, and (ii) even when the adversary receives simulated proofs in the left sessions, witnesses can be extracted from the adversary in the right sessions. In the protocol of [BPS06, LPTV10], the simulatability of the left sessions is guaranteed by requiring the verifier to commit to a random trapdoor by using the *concurrently extractable commitment scheme* CECom of Micciancio et al. [MOSV06]. The committed values of CECom can be extracted by a rewinding extractor even in the concurrent setting, so the proofs in the left sessions can be simulated by extracting the trapdoors from CECom. On the other hand, the witness-extractability of the right sessions is guaranteed by requiring the prover to commit to the witness by using a non-malleable (NM) commitment scheme [DDN00] so that the following hold.

1. When the adversary receives honest proofs in the left sessions, the committed value of the NM commitments in each accepted right session is indeed a valid witness.

2. When the proofs in the left sessions are switched to the simulated ones, the committed values of the NM commitments in the right sessions do not change due to their non-malleability.

It follows from these two that even when the adversary receives simulated proofs in the left sessions, the committed value of the NM commitment is a witness for the statement in every accepted right session. Therefore, the witnesses can be extracted in the right sessions by extracting the committed values of the NM commitments.

As mentioned in the introduction, the techniques of [BPS06, LPTV10] alone seem to be insufficient for constructing statistical CNMZK protocols. The main obstacle is that the techniques of [BPS06, LPTV10] require the prover to commit to the witness by using a NM commitment scheme, which is only computationally hiding.[10] Since the committed values of the NM commitments in the left sessions need to be switched to other values (e.g., 0^n) in the simulation, the simulated view can be only computational indistinguishable from the real view.

[10] The NM commitment scheme used here need to be *non-malleable w.r.t. commitment* [DDN00], which roughly says that the committed value of the commitment that the man-in-the-middle adversary gives is independent of the committed value of the commitment that adversary receives. Since the definition of non-malleability w.r.t. commitment is meaningless when committed values cannot be uniquely determined, the NM commitment scheme used here cannot be statistically hiding.

Recently, Orlandi et al. [OOR+14] constructed a statistical CNMZK protocol by modifying the CNMZK protocol of [BPS06] with a *mixed non-malleable commitment scheme*. A mixed NM commitment scheme is parametrized by a string and is either statistically hiding or non-malleable depending on the parameter string.[11] Very roughly speaking, Orlandi et al. circumvent the above problem by carefully switching the parameter string of the mixed NM commitment scheme in the security proof—when proving the statistical indistinguishability of the simulation, the string is set so that the mixed NM commitment scheme is statistically hiding, and when proving the non-malleability, the string is set so that it is non-malleable. The use of a mixed NM commitment scheme, however, requires assumptions that are seemingly stronger than the existence of one-way functions (such as the DDH assumption or the existence of dense cryptosytems). Thus, the technique of Orlandi et al. cannot be used to construct statistical CNMZK protocols from one-way functions.

3.2.2 Our Techniques

Since the reason why the techniques of [BPS06, LPTV10] cannot be used for statistical CNMZK protocols is that the committed values of the NM commitments need to be switched during the simulation, one potential strategy for constructing statistical CNMZK is to construct a protocol such that the adversary's view can be simulated without switching the committed values of the NM commitments (and of any other computationally hiding commitments). However, when the simulator commits to the same values in the NM commitments as an honest prover, it is not clear how their non-malleability can be used in the security proof. Roughly speaking, we show that the CNMZK property can be shown even in this case if we use a stronger variant of NM commitment schemes.

A key technical tool in our technique is *CCA-secure commitment schemes* [CLP10, CLP16], which are a stronger variant of (concurrent) non-malleable commitment schemes. Roughly speaking, CCA security guarantees that the scheme is hiding even against adversaries that have access to the *decommitment oracle*, which takes concurrent commitments from the adversary and returns their decommitments (which are computed by brute force by the oracle) to the adversary. Several CCA-secure commitment schemes were constructed from one-way functions [CLP10, LP12, Kiy14, GLP+15];[12] furthermore, although CCA security itself does not provide any extractability, all of these schemes satisfy concurrent extractability as well.

Using CCA-secure commitment schemes, we consider the following protocol as a starting point.

Stage 1. (V commits to trapdoor) The verifier V chooses random $r_V \in \{0,1\}^n$ and commits to r_V by using $\mathsf{CCA\text{-}CECom}$, where $\mathsf{CCA\text{-}CECom}$ is a CCA-secure

[11] Specifically, Orlandi et al. [OOR+14] used a scheme such that (i) when the string is sampled from a uniform distribution, it is statistically hiding and (ii) when the string is taken from another (computationally indistinguishable) distribution, it is non-malleable.

[12] Actually, some of these constructions (namely, those by [LP12, Kiy14]) satisfy only a sightly weaker notion called *CCA security w.r.t. the committed-value oracle*.

commitment scheme that is also concurrent extractable. For simplicity, we assume that the extractor of CCA-CECom extracts a decommitment rather than the committed value.

Stage 2. (P proves $x \in L$ or knowledge of trapdoor) The prover P proves that it knows a witness for $x \in L$ or a valid decommitment (r_V, d) for the CCA-CECom commitment that V gives in Stage 1. P proves this statement by using a statistical witness-indistinguishable special-sound argument of knowledge sWIAOK, which can be constructed from one-way functions by instantiating Blum's Hamiltonian-cycle protocol with the statistically hiding commitment scheme of [HNO+09]. For concreteness, we assume that we indeed obtain sWIAOK in this way.

In this protocol, the verifier's view can be statistically simulated by a simulator that extracts (r_V, d) from CCA-CECom and uses it as a witness in sWIAOK. (Recall that we assume for simplicity that the extractor of CCA-CECom extracts a decommitment rather than the committed value.) During the extraction in Stage 1, the simulator interacts with the verifier honestly; thus, if computationally hiding commitment schemes are used as building blocks in CCA-CECom, the simulator commits to the same values as an honest prover in these schemes as required.

Intuitively, this protocol is CNMZK from the following reasons.

- The CCA security of CCA-CECom guarantees that the CCA-CECom commitments in the right sessions are hiding even when the adversary receives simulated proofs in the left sessions. This is because the simulated proofs in the left sessions can be generated efficiently given decommitments for the CCA-secure commitments of the left sessions, and the CCA security of CCA-CECom guarantees that the CCA-CECom commitments in the right sessions are hiding even when the adversary receives decommitments for those CCA-CECom commitments.

- Thus, even when the adversary receives simulated proofs in the left sessions, the adversary cannot "cheat" in the right sessions, so witnesses for the statements in the right sessions must be extractable from their sWIAOK proofs.

Of course, to formally prove the statistical CNMZK property, we need to show a simulator-extractor that statistically simulates the adversary's view and also extracts witnesses for the statements in the right sessions.

As the simulator-extractor, we consider the following $\mathcal{SE}$.

1. First, $\mathcal{SE}$ simulates the view of the adversary $\mathcal{A}$ by executing the following simulator $\mathcal{S}$: Simulator $\mathcal{S}$ internally invokes $\mathcal{A}$ and interacts with it in the left and right sessions honestly as provers and verifiers except that in each left session, $\mathcal{S}$ extracts (r_V, d) by using the concurrent extractor of CCA-CECom and uses it as a witness in sWIAOK.

2. After simulating the view of $\mathcal{A}$ as above, $\mathcal{SE}$ extracts witnesses from the right sessions by doing the following for each right session. First, $\mathcal{SE}$ rewinds $\mathcal{S}$ until

the point just before $\mathcal{S}$ sends the challenge message of sWIAOK to $\mathcal{A}$.[13] Then, $\mathcal{SE}$ repeatedly executes $\mathcal{S}$ from this point with fresh randomness until it obtains another accepted transcript of sWIAOK. After obtaining another accepted transcript, $\mathcal{SE}$ extracts a witness by using the special soundness of sWIAOK.

It is not hard to see that $\mathcal{SE}$ statistically simulates the real view of $\mathcal{A}$. Thus, it remains to show that $\mathcal{SE}$ extracts witnesses for the statements in the right sessions.

To show the witness extractability of $\mathcal{SE}$, a natural approach is to follow the above-mentioned approach of [BPS06, LPTV10] and show the following.

1. When $\mathcal{A}$ receives honest proofs in the left sessions, a witness for the statement is extracted from the sWIAOK proof in every accepted right session.

2. When the honest proofs in the left sessions are switched to the simulated ones, the value extracted from sWIAOK does not change in every accepted right session.

Note that here we need to argue about the extracted values instead of the committed values. At first sight, this does not seem to be a big difference, and it seems that the above can be shown by using an argument similar to the one used in [BPS06, LPTV10].

However, this approach does not work. In particular, we do not know how to prove the second part—that is, we cannot show that the extracted values remain to be the same when the honest proofs in the left sessions are switched to the simulated ones. To see this, observe the following. Since the witnesses used in sWIAOK are switched in the simulated proofs, we need to use the witness indistinguishability of the sWIAOK proofs of the left sessions to show the indistinguishability of the extracted values. However, since $\mathcal{A}$ is rewound during the witness extraction of the sWIAOK proofs of the right sessions, if the left and the right sessions are scheduled so that the sWIAOK proofs of the left sessions are executed in parallel with the sWIAOK proofs of the right sessions, the sWIAOK proofs of the left sessions are also rewound, and we cannot use their witness indistinguishability.[14]

Thus, we instead use the following approach. Informally, the above approach does not work because the honest proofs and the simulated proofs are "too different." We thus introduce a hybrid experiment in which $\mathcal{A}$ receives *hybrid proofs* in the left sessions, where a hybrid proof is generated by extracting (r_V, d) by brute force and using it as a witness in sWIAOK. (Notice that the only difference between the hybrid proofs and the simulated proofs is how the trapdoors are extracted.) We then show that (i) witnesses for the statements in the right sessions are extracted when $\mathcal{A}$ receives hybrid proofs in the left sessions, and (ii) when hybrid proofs are switched to the simulated ones, the extracted values do not change. More precisely, our analysis proceeds as follows.

[13] Since $\mathcal{S}$ rewinds $\mathcal{A}$ during the concurrent extraction of CCA-CECom, $\mathcal{S}$ may send the challenge message of sWIAOK of a right session to $\mathcal{A}$ multiple times. Here, $\mathcal{SE}$ rewinds $\mathcal{S}$ until the point just before $\mathcal{S}$ sends it to $\mathcal{A}$ on the "main thread."

[14] If we use the robust extraction technique [GLP+15], for each left session there exists a rewinding strategy that allows us to extract witnesses from the right sessions without rewinding sWIAOK of this left session. However, since what we want to show is that the values extracted in the right sessions *by the rewinding strategy that $\mathcal{SE}$ uses* are unchanged, the robust extraction technique cannot be used here (unless there exists a rewinding strategy that allows us to extract witnesses from the right sessions without rewinding the sWIAOK proof of *every* left session).

- First, we show the second part, i.e., we show that the values extracted in the right sessions do not change when the proofs in the left sessions are switched from the hybrid proofs to the simulated ones. Since the only difference between the hybrid proofs and the simulated ones is how the committed values of the CCA-CECom commitments are extracted (by brute-force or by the concurrent extractability), we can show this by using the concurrent extractability of CCA-CECom. We note however that there is a subtlety since CCA-CECom in the left sessions can be rewound not only by the concurrent extractor of CCA-CECom but also by the extractor of sWIAOK. Nonetheless, by carefully using a standard technique (the "good prefix" argument), we can show that the concurrent extractor of CCA-CECom works even in this case.

- Next, we show the first part, i.e., we show that witnesses for the statements in the right sessions are extracted in the hybrid experiment. Since the simulated proofs can be efficiently generated given access to the decommitment oracle of CCA-CECom, at first sight it seems that this follows directly from the CCA security of CCA-CECom and argument-of-knowledge property of sWIAOK—if a witness for the statement is not extracted in a right session, a valid decommitment for the CCA-CECom commitment must be extracted in that right session, so we can break the CCA security of CCA-CECom. However, there are two problems.

  1. Since CCA-CECom in the left sessions can be rewound during the witness extraction of sWIAOK of the right sessions, the hybrid experiment cannot be emulated even given access to the decommitment oracle of CCA-CECom. Hence, the CCA-secure commitments in the right sessions may not be hiding in the hybrid experiment.
  2. Since the adversary obtains hybrid proofs, which are generated in super-polynomial time, the argument-of-knowledge property of sWIAOK may not hold in the hybrid experiment. We remark that although existing CCA-secure commitment schemes provide *robustness*, which guarantees that arbitrary "small"-round protocol remains secure even when adversaries have access to the decommitment oracle, we cannot use robustness here since CCA-CECom in the left sessions can be rewound during the witness extraction of sWIAOK of the right sessions and therefore the hybrid experiment cannot be emulated even given access to the decommitment oracle.

  Because of these problems, we cannot use the security of CCA-CECom in a modular way in the analysis. Thus, we directly use the building blocks of existing CCA-secure commitment schemes in the actual construction of our protocol, and use their proof techniques in the analysis. The proof techniques of existing CCA-secure commitment schemes are strong enough to solve the above problems, so we can show that witnesses for the statements are extracted in the hybrid experiment. (In a bit more detail, we use a specific CCA-secure commitment scheme as CCA-CECom in the actual construction of our protocol, where we obtain this CCA-secure commitment scheme by using *one-one CCA-secure commitment schemes* [KMO14] and the *robust concurrent extraction technique*

[GLP+15]. In the analysis, we inline the proof of CCA security and robustness of this CCA-secure commitment scheme and then observe that its CCA security and robustness hold even in the presence of the witness extraction of sWIAOK of the right sessions.)

From the above two, it follows that even when $\mathcal{A}$ receives simulated proofs in the left session, valid witnesses are extracted in right sessions.

The formal description of our protocol can be found in Figure 3.4 in Section 3.2. The second part of our analysis (i.e., the part where we show that the values extracted in the right sessions do not change when the proofs in the left sessions are switched from the hybrid proofs to the simulated ones) is described in Section 3.4.2.2. The first part of our analysis (i.e., the part where we show that witnesses for the statements are extracted from the right sessions in the hybrid experiment) is described in Section 3.4.2.3.

3.3 Preliminaries

3.3.1 Concurrently Extractable Commitment Schemes

In this section, we explain the concurrently extractable commitment scheme that we use in this chapter. (Recall that the notion of concurrently extractable commitment scheme is explained in Section 2.3.3.) We notice that concurrently extractable commitment schemes are used in this chapter in a "non-black-box" way—that is, we directly use a specific construction and extractor in our protocol and analysis rather than using them in a modular way.)

The concurrently extractable commitment scheme that we use is the one by Micciancio et al. [MOSV06], which we denote by CECom and describe in Figure 2.2. As explained in Section 2.3.3, CECom is constructed from one-way functions and has round complexity $\omega(\log n)$. Also, CECom is an abstraction of the preamble stage of the concurrent zero-knowledge protocol of Prabhakaran et al. [PRS02], and its extractor obtains committed values of concurrent CECom commitments by rewinding the committer according to the recursive rewinding strategy of Prabhakaran et al. When using the rewinding strategy of Prabhakaran et al., the extractor honestly interacts with the adversarial committer C^* on the "main thread" as an honest receiver while rewinding C^* and generating many "look-ahead threads" on which it interacts with C^* again as an honest receiver by using fresh randomness (see Figure 3.1). It is guaranteed that whenever C^* completes a session of the commit phase on the main thread or on a look-ahead thread, the extractor can compute the committed value of that session by using the information collected so far on the look-ahead threads. More precisely, it is guaranteed that if C^* gives a valid commitment on a thread, the extractor can output its correct committed value at the end of the commitment except with negligible probability; when C^* gives an invalid commitment, there is no guarantee about the value that the extractor outputs at the end of the commitment, and the extractor can output an arbitrary value as the committed value.

The extractor that we use for CECom is the one by Goyal et al. [GLP+15]. The extractor by Goyal et al. uses a rewinding strategy that is based of that of Prabhakaran et

al.; thus, it interacts with the adversarial committer on the main thread honestly while interacting with C^* again on the look-ahead threads with fresh randomness. A nice property of the extractor by Goyal et al. [GLP+15] is *robust concurrent extraction* (which is formalized as the robust concurrent extraction lemma in Section 2.3.3.1). Roughly speaking, this property guarantees that even when the adversarial committer additionally participates in an external "small"-round protocol, committed values can be extracted from concurrent commitments *without rewinding the external protocol*. More precisely, consider any PPT adversarial committer $\mathcal{A}$ that commits to multiple values in concurrent sessions of CECom—these sessions are denoted as the *right sessions*—and simultaneously participates in an execution of an arbitrary protocol $\Pi := \langle B, A \rangle$ with an honest B—this session is denoted as the *left session* (see Figure 3.2). The robust concurrent extraction property guarantees that for every such $\mathcal{A}$, there exists an extractor E that extracts committed values from $\mathcal{A}$ in all the valid right sessions (that is, whenever $\mathcal{A}$ completes a right session on the main thread or on one of the look-ahead threads, the extractor can compute the committed value of that session as long as that session is valid), and E does not rewind the external party B during the extraction. The extractor E fails with probability that is exponentially small in $\ell - O(k \log n)$, where ℓ is the parameter of CECom and k is the round complexity of Π; hence, E fails only with negligible probability if we set $\ell := \omega(k \log n)$. For a formal description of the extractor by Goyal et al. (which is not necessary to understand this chapter), see [GLP+15].

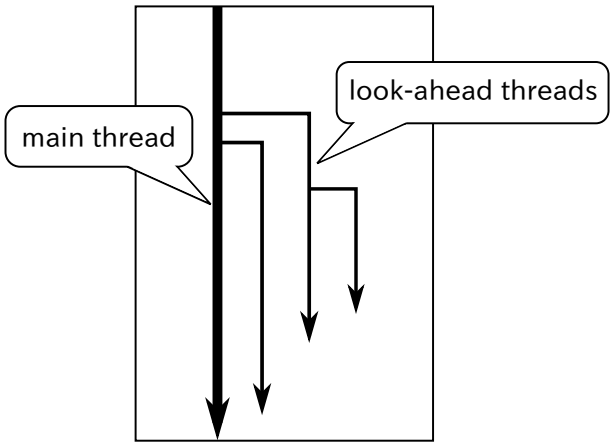

Figure 3.1: Main thread and look-ahead threads.

Figure 3.2: Robust concurrent adversary.

3.3.2 One-one CCA-secure Commitment Schemes

In this section, we explain the construction of one-one CCA-secure commitment scheme that that we use in this chapter. (Recall that the definition of one-one CCA-secure commitment schemes is given in Section 2.3.5.) From a result shown in [GLP+15], we can obtain a constant-round κ-robust one-one CCA-secure commitment scheme from one-way functions for every constant $\kappa \in \mathbb{N}$ as follows. In [GLP+15], Goyal et al. constructed a $\omega(\log n)$-round CCA-secure commitment scheme from one-way functions. This scheme has $\omega(\log n)$ rounds because CECom with parameter $\ell = \omega(\log n)$ is used as a building block, where CECom is the concurrently extractable commitment scheme of Micciancio et al. [MOSV06] (see Section 3.3.1). The reason why ℓ is set to be $\omega(\log n)$ is that in the security analysis, the committed values of CECom need to be extracted when polynomially many CECom commitments are concurrently executed. In the setting of *one-one* CCA security, however, the security analysis works even if the committed values of CECom are extractable only when a single CECom commitment is executed. Hence, by setting $\ell := O(1)$, we can obtain a constant-round one-one CCA-secure commitment scheme. For completeness, we give the protocol and the proof of one-one CCA security in Section 3.5.1.

3.3.3 Witness Indistinguishable Proofs and Arguments

In this section, we explain the witness-indistinguishable proof/argument of knowledge systems that we use in this chapter. (Recall that the definitions of witness-indistinguishable proofs/arguments of knowledge are given in Section 2.4.) A four-round witness-indistinguishable proof of knowledge system can be obtained from one-way functions by instantiating (a parallel version of) Blum's Hamiltonian-cycle protocol with Naor's commitment scheme (cf. Section 2.4.2). Also, a statistical witness-indistinguishable argument of knowledge system can be obtained from any statistical hiding commitment scheme by instantiating Blum's Hamiltonian-cycle protocol with a statistically hiding commitment scheme. This argument system satisfies *special soundness* in the following sense: Let us say that two accepting transcripts $\langle \vec{\alpha}_1, \beta_1, \gamma_1 \rangle$ and $\langle \vec{\alpha}_2, \beta_2, \gamma_2 \rangle$ are *admissible* if $\vec{\alpha}_1 = \vec{\alpha}_2$ and $\beta_1 \neq \beta_2$ (where $\vec{\alpha}_1, \vec{\alpha}_2$ are the commitments from the prover and β_1, β_2 are the challenges from the verifier); then, given a pair of admissible transcripts that are generated in polynomial time, one can compute a valid witness. In particular, given admissible transcripts $\langle \vec{\alpha}, \beta_1, \gamma_1 \rangle$ and $\langle \vec{\alpha}, \beta_2, \gamma_2 \rangle$, either one can compute a valid witness or one can decommit a commitment given in $\vec{\alpha}$ to two different values.

3.3.4 Statistical Concurrent Non-malleable Zero-knowledge Arguments

In this section, we recall the definition of (statistical) concurrent non-malleable zero-knowledge from [BPS06, OOR+14], which is closely related to the definition of simulation extractability of [PR05b, PR08]. Let $\langle P, V \rangle$ be an interactive argument system for a language $L \in \mathcal{NP}$ with witness relation $\mathbf{R}_L$. For any man-in-the-middle adversary $\mathcal{A}$,

let us consider a probabilistic experiment in which $\mathcal{A}$ participates in the following left and right interactions (see Figure 3.3). In the left interaction, $\mathcal{A}$ interacts with an honest prover P of $\langle P, V\rangle$ and verifies the validity of statements $x_1, \ldots, x_m$ using identities $\mathsf{id}_1, \ldots, \mathsf{id}_m$. In the right interaction, $\mathcal{A}$ interacts with an honest verifier V of $\langle P, V\rangle$ and proves the validity of statements $\widetilde{x}_1, \ldots, \widetilde{x}_m$ using identities $\widetilde{\mathsf{id}}_1, \ldots, \widetilde{\mathsf{id}}_m$. The statements proven in the left interaction, $x_1, \ldots, x_m$, are given to P and $\mathcal{A}$ prior to the experiment. In contrast, the statements proven in the right interaction, $\widetilde{x}_1, \ldots, \widetilde{x}_m$, and the identities used in the left and the right interactions, $\mathsf{id}_1, \ldots, \mathsf{id}_m$ and $\widetilde{\mathsf{id}}_1, \ldots, \widetilde{\mathsf{id}}_m$, are chosen by $\mathcal{A}$ during the experiment. Let $\mathsf{view}_{\mathcal{A}}(n, x_1, \ldots, x_m, z)$ be a random variable representing the view of $\mathcal{A}$ in the above experiment. Then, roughly speaking, $\langle P, V\rangle$ is *statistical concurrent non-malleable zero-knowledge* (statistical CNMZK) if for any adversary $\mathcal{A}$, there exists a PPT machine called the *simulator-extractor* that can statistically simulate the view of $\mathcal{A}$ in the above experiment while extracting witnesses for the statements proven by $\mathcal{A}$ in the accepted right interactions that use different identities from the left interactions. The formal definition is given below.

Definition 3.1. *An interactive proof $\langle P, V\rangle$ for language L with witness relation $\boldsymbol{R}_L$ is said to be **statistical concurrent non-malleable zero-knowledge** if for every polynomial $m(\cdot)$ and every probabilistic polynomial-time man-in-the-middle adversary $\mathcal{A}$ that participates in at most $m = m(n)$ concurrent executions, there exists a probabilistic polynomial-time machine $\mathcal{SE}$ (called a **simulator-extractor**) such that the following hold.*

1. *Let $\mathsf{sim\text{-}view}(n, x_1, \ldots, x_m, z)$ be a random variable representing the first output of $\mathcal{SE}(n, x_1, \ldots, x_m, z)$. Then, the following ensembles are statistically indistinguishable.*
   - $\{\mathsf{view}_{\mathcal{A}}(n, x_1, \ldots, x_m, z)\}_{n\in\mathbb{N}, x_1, \ldots, x_m \in L\cap\{0,1\}^n, z\in\{0,1\}^*}$
   - $\{\mathsf{sim\text{-}view}(n, x_1, \ldots, x_m, z)\}_{n\in\mathbb{N}, x_1, \ldots, x_m \in L\cap\{0,1\}^n, z\in\{0,1\}^*}$
2. *For every $n \in \mathbb{N}$, $x_1, \ldots, x_m \in L \cap \{0,1\}^n$, and $z \in \{0,1\}^*$, the following holds. Let $(\mathsf{view}, \{\widetilde{w}_i\}_{i\in[m]})$ denote the output of $\mathcal{SE}(n, x_1, \ldots, x_m, z)$. Let $\widetilde{x}_1, \ldots, \widetilde{x}_m$ be the statements of the right interaction in view, and let $\mathsf{id}_1, \ldots, \mathsf{id}_m$ and $\widetilde{\mathsf{id}}_1, \ldots, \widetilde{\mathsf{id}}_m$ be the identities of the left and the right interactions in view, respectively. Then, except with negligible probability, we have $(\widetilde{x}_i, \widetilde{w}_i) \in \boldsymbol{R}_L$ for every $i \in [m]$ such that the i-th right interaction is accepting and $\widetilde{\mathsf{id}}_i \neq \mathsf{id}_j$ holds for every $j \in [m]$.* ◇

3.4 Our Statistical Concurrent Non-malleable ZK Argument

In this section, we show that a statistical concurrent non-malleable zero-knowledge argument can be constructed from any statistically hiding commitment scheme.

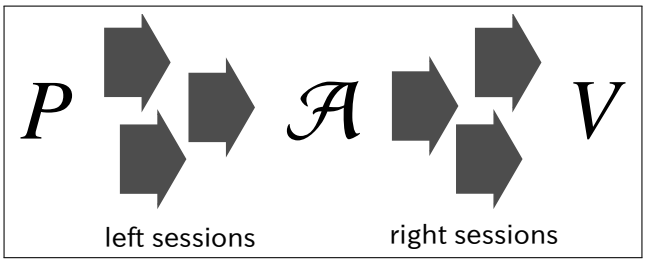

Figure 3.3: Left sessions and right sessions.

Theorem 3.2. *Assume the existence of statistically hiding commitment schemes with round complexity $R_{\mathsf{SH}}(n)$. Then, there exists a $\omega(R_{\mathsf{SH}}(n)\log n)$-round statistical concurrent non-malleable zero-knowledge argument* sCNMZK.

Since a $\mathsf{poly}(n)$-round statistically hiding commitment scheme can be constructed from one-way functions [HNO+09] and a constant-round one can be constructed from a family of collision-resistant hash functions [NY89, DPP98], our main theorem (Theorem 3.1) follows from Theorem 3.2.

Proof of Theorem 3.2. In sCNMZK, we use the following building blocks.

- Two-round statistically binding commitment scheme $\mathsf{Com}_{\mathsf{SB}}$.
- Constant-round 4-robust one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$.
- Four-round witness-indistinguishable proof WIProof, which is a parallel version of Blum's Hamiltonian-cycle protocol.
- $(R_{\mathsf{SH}}(n)+2)$-round statistical witness-indistinguishable argument of knowledge sWIAOK, which is a parallel version of Blum's Hamiltonian-cycle protocol that is instantiated with a $R_{\mathsf{SH}}(n)$-round statistically hiding commitment scheme $\mathsf{Com}_{\mathsf{SH}}$.
- $\omega(R_{\mathsf{SH}}(n)\log n)$-round concurrently extractable commitment scheme CECom, which is the scheme by Micciancio et al. [MOSV06] with parameter $\ell = \omega(R_{\mathsf{SH}}(n)\log n)$. By using the extractor by Goyal et al. [GLP+15], we can extract committed values from any adversarial committer even when it additionally participates in any $O(R_{\mathsf{SH}}(n))$-round external protocol (see Section 3.3.1).

As explained in Sections 2.3.1 and 3.3, all of the above building blocks can be constructed from $R_{\mathsf{SH}}(n)$-round statistically hiding commitment schemes (or from one-way functions, which can be obtained from statistically hiding commitment schemes).

Protocol sCNMZK is shown in Figure 3.4. We prove its soundness in Section 3.4.1 and prove its statistical CNMZK property in Section 3.4.2.

3.4.1 Proof of Soundness

Lemma 3.1. *Protocol* sCNMZK *is sound.*

Input. The common input is statement $x \in L$ and identity $\mathsf{id} \in \{0,1\}^n$. The prover's private input is witness $w \in \mathbf{R}_L(x)$.

Stage I. (V commits to trapdoor)

1. V chooses random $r_V \in \{0,1\}^n$ and commits to r_V by using $\mathsf{Com}_{\mathsf{SB}}$. Let (r_V, d) be the decommitment of this commitment.
2. V commits to (r_V, d) by using CECom.
3. P chooses random $r_P \in \{0,1\}^n$ and commits to r_P by using $\mathsf{CCACom}^{1:1}$ with tag id.
4. V commits to 0^n by using CECom.
5. P decommits the $\mathsf{CCACom}^{1:1}$ commitment in Stage I-3 to r_P.
6. V proves the following by using $\mathsf{WIProof}$:
   - the committed value of the CECom commitment in Stage I-2 is a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1, or
   - the committed value of the CECom commitment in Stage I-4 is r_P.

COMMENT: *Essentially, what V does in this stage is to commit to r_V by using a specific CCA-secure commitment scheme as explained in Section 3.2.2. This CCA-secure commitment scheme has the commit-and-proof structure: V commits to r_V in Stages I-1 – I-2, and proves the correctness of this commitment in Stages I-3 – I-6.*

Stage II. (P proves $x \in L$ or knowledge of trapdoor)

1. P proves the following by using sWIAOK:
   - $x \in L$, or
   - there exists (r'_V, d') such that (r'_V, d') is a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1.

Figure 3.4: Statistical concurrent non-malleable zero-knowledge argument sCNMZK.

Proof. Assume for contradiction that there exists an adversarial prover P^* that breaks the soundness of sCNMZK. It follows from the argument-of-knowledge property of sWIAOK and the binding property of $\mathsf{Com}_{\mathsf{SB}}$ that we can extract r_V from P^* in Stage II with non-negligible probability, where r_V is the value committed to by the verifier in Stage I-1. In the following, we consider a sequence of hybrid experiments in which the verifier is gradually modified so that P^* receives no information about r_V in the last hybrid, and then we derive a contradiction by showing that r_V is still extractable with non-negligible probability in the last hybrid.

Hybrid H_0 is an experiment in which an honest verifier interacts with P^* and then a witness is extracted from P^* in Stage II by the knowledge extractor of sWIAOK. The output of H_0 is the witness extracted in Stage II. From the above observation, the output of H_0 is r_V with non-negligible probability.

Hybrid H_1 is the same as H_0 except that (i) the committed value r_P of the $\mathsf{CCACom}^{1:1}$ commitment in Stage I-3 is extracted by the one-session committed-value oracle O of $\mathsf{CCACom}^{1:1}$ and (ii) the committed value of the CECom commitment in Stage I-4 is switched from 0^n to r_P.

Note that, basically, the only difference between H_0 and H_1 is the value committed to by using CECom in Stage I-4. However, since the execution of H_1 involves a super-polynomial-time computation (i.e., the extraction of r_P), we cannot directly use the hiding property of CECom to argue that the output of H_1 is indistinguishable from that of H_0. Nevertheless, since H_1 can be executed in polynomial-time given access to the one-session committed-value oracle of $\mathsf{CCACom}^{1:1}$, we can show the indistinguishability between the output of H_1 and that of H_0 by combining the hiding property of CECom with the robustness of $\mathsf{CCACom}^{1:1}$ (cf. Proposition 2.1 in Section 2.3.5).
Comment: *Formally, since* $\mathsf{CCACom}^{1:1}$ *is robust only w.r.t. 4-round protocols, we need to consider a sequence of intermediate hybrids in which the* CECom *commitment is gradually modified by switching the committed values of the* ExtCom *commitments one by one in* CECom *(cf. Figure 2.2 in Section 2.3.3). Since* ExtCom *has only four rounds, the 4-robustness of* $\mathsf{CCACom}^{1:1}$ *guarantees that the outputs of these intermediate hybrids are indistinguishable.*

Hybrid H_2 is the same as H_1 except that the $\mathsf{WIProof}$ proof in Stage I-6 is computed by using a witness for the fact that the committed value of the CECom commitment in Stage I-4 is r_P.

Similar to the above, the indistinguishability between the output of H_2 and that of H_1 follows from the witness indistinguishability of $\mathsf{WIProof}$ and the robustness of $\mathsf{CCACom}^{1:1}$.

Hybrid H_3 is the same as H_2 except that in Stage I-2, the committed value of the CECom commitment is switched from (r_V, d) to $(0^{|r_V|}, 0^{|d|})$.

The indistinguishability between the output of H_3 and H_2 follows from the hiding property of CECom (or, more precisely, the hiding property of ExtCom used in CECom) and the robustness of $\mathsf{CCACom}^{1:1}$.

Hybrid H_4 is the same as H_3 except that in Stage I-1, the committed value of the $\mathsf{Com}_{\mathsf{SB}}$ commitment is switched from r_V to 0^n.

The indistinguishability between the output of H_4 and that of H_3 follows from the hiding property of $\mathsf{Com}_{\mathsf{SB}}$ and the robustness of $\mathsf{CCACom}^{1:1}$.

From the above, the probability that the output of H_4 is r_V is non-negligible. However, since P^* receives no information about r_V in H_4, this probability must be negligible. Thus, we reach a contradiction. □

3.4.2 Proof of Statistical CNMZK Property

3.4.2.1 Simulator-extractor $\mathcal{SE}$.

Recall that to prove the statistical CNMZK property, we need to show a simulator-extractor that statistically simulates the view of the adversary $\mathcal{A}$ while extracting a witness in every accepted right session. We construct our simulator-extractor step by step. First, we construct a super-polynomial-time simulator $\hat{\mathcal{S}}$ that simulates the view of $\mathcal{A}$ but does not extract witnesses in the right sessions. Next, we construct a super-polynomial-time simulator-extractor $\hat{\mathcal{SE}}$ that simulates the view of $\mathcal{A}$ by executing $\hat{\mathcal{S}}$ and then extracts witnesses by rewinding $\hat{\mathcal{S}}$. Finally, we construct a polynomial-time simulator-extractor $\mathcal{SE}$ that emulates the execution of $\hat{\mathcal{SE}}$ in polynomial time.

Remark 3.1. In the following, we use the hat symbol in the names of simulators and simulator-extractors if they run in super-polynomial time (e.g., $\hat{\mathcal{S}}$ and $\hat{\mathcal{SE}}$).

Remark 3.2. In the following, we use the tilde symbol in the names of the messages of sCNMZK if they are the messages of the right sessions (e.g., $\widetilde{r}_V$ and $\widetilde{r}_P$). If necessary, we use subscript to denote the index of the session.

Super-polynomial-time simulator $\hat{\mathcal{S}}$. First, the simulator $\hat{\mathcal{S}}$ simulates the view of $\mathcal{A}$ in super-polynomial time as follows. $\hat{\mathcal{S}}$ internally invokes $\mathcal{A}$ and interacts with $\mathcal{A}$ as provers and verifiers in the following way.

- In each left session, $\hat{\mathcal{S}}$ interacts with $\mathcal{A}$ in the same way as an honest prover except for the following. In Stage I-2, $\hat{\mathcal{S}}$ extracts the committed value (r_V, d) of the CECom commitment by brute force. (If the committed value is not uniquely determined, (r_V, d) is defined to be $(\bot, \bot)$.) In Stage II, $\hat{\mathcal{S}}$ checks whether (r_V, d) is a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1; if so, $\hat{\mathcal{S}}$ gives a sWIAOK proof by using (r_V, d) as a witness; otherwise, $\hat{\mathcal{S}}$ terminates with output fail.

- In each right session, $\hat{\mathcal{S}}$ interacts with $\mathcal{A}$ in the same way as an honest verifier.

Finally, $\hat{\mathcal{S}}$ outputs the view of internal $\mathcal{A}$. Notice that $\hat{\mathcal{S}}$ does not rewind $\mathcal{A}$.

Super-polynomial-time simulator-extractor $\hat{\mathcal{SE}}$. Next, the simulator-extractor $\hat{\mathcal{SE}}$ simulates the view of $\mathcal{A}$ in super-polynomial time and extracts witnesses in the accepted right sessions as follows. First, $\hat{\mathcal{SE}}$ simulates the view of $\mathcal{A}$ by executing $\hat{\mathcal{S}}$. We call this execution of $\hat{\mathcal{S}}$ the *wi-main thread*. Next, for each $i \in [m]$, if the i-th right session is accepted on the wi-main thread and uses a different identity from every left session, $\hat{\mathcal{SE}}$ extracts a witness from this session as follows.

- $\hat{\mathcal{SE}}$ rewinds the wi-main thread until the point just before the challenge message of sWIAOK of the i-th right session is sent. Then, from this point, $\hat{\mathcal{SE}}$ executes $\hat{\mathcal{S}}$ again with fresh randomness (i.e., interacts with $\mathcal{A}$ as $\hat{\mathcal{S}}$ does with fresh randomness). $\hat{\mathcal{SE}}$ repeats this rewinding until it obtains another accepting transcript of the i-th right session. We call each execution of $\hat{\mathcal{S}}$ in this step a *wi-auxiliary thread*.

- After obtaining two accepting transcripts of the i-th right session (one is on the wi-main thread and the other is on an wi-auxiliary thread), $\hat{\mathcal{SE}}$ extracts a witness from sWIAOK by using the special soundness of sWIAOK. If $\hat{\mathcal{SE}}$ fails to extract a witness for $\widetilde{x}_i \in L$ (the statement proven in the i-th right session), $\hat{\mathcal{SE}}$ terminates with output $\mathsf{fail}_{\mathsf{WI}}$. Otherwise, let $\widetilde{w}_i$ be the extracted witness.

If the i-th right session is not accepted or uses the same identity as a left session, define $\widetilde{w}_i \stackrel{\text{def}}{=} \bot$. The output of $\hat{\mathcal{SE}}$ is $(\mathsf{view}, \{\widetilde{w}_i\}_{i\in[m]})$, where view is the view of $\mathcal{A}$ on the wi-main thread.

Polynomial-time simulator-extractor $\mathcal{SE}$. Finally, the simulator-extractor $\mathcal{SE}$ emulates the execution of $\hat{\mathcal{SE}}$ in polynomial time as follows. First, $\mathcal{SE}$ emulates the wi-main thread in polynomial time as follows.

- $\mathcal{SE}$ internally invokes $\mathcal{A}$ and interacts with $\mathcal{A}$ as $\hat{\mathcal{S}}$ does except that in each left session, $\mathcal{SE}$ extracts (r_V, d) by using the concurrent extractability of CECom instead of by brute force. Recall that a concurrent extraction of CECom involves the generation of a main thread and many look-ahead threads (see Section 3.3.1). We call the main thread generated during the concurrent extraction of CECom the *cec-main thread*, and call the look-ahead threads generated during the concurrent extraction of CECom the *cec-auxiliary threads*.[15] (See Figure 3.5.)

Next, for each $i \in [m]$, if the i-th right session is accepted on the emulated wi-main thread and uses a different identity from every left session, $\mathcal{SE}$ emulates wi-auxiliary threads as follows.

- $\mathcal{SE}$ rewinds the emulation of the wi-main thread until the point just before the challenge message of sWIAOK of the i-th right session is sent on the cec-main thread. Then, from this point, $\hat{\mathcal{SE}}$ emulates the wi-main thread again with fresh randomness (i.e., generates the rest of cec-main thread and cec-auxiliary threads with fresh randomness). $\mathcal{SE}$ repeats this rewinding until it obtains another accepted transcript of the i-th right session on an emulated wi-auxiliary thread.

Let $(\mathsf{view}, \{\widetilde{w}_i\}_{i\in[m]})$ be the output of the emulated $\hat{\mathcal{SE}}$. Then, $\mathcal{SE}$ outputs $(\mathsf{view}, \{\widetilde{w}_i\}_{i\in[m]})$.

[15] Note that the wi-main thread is also a cec-main thread.

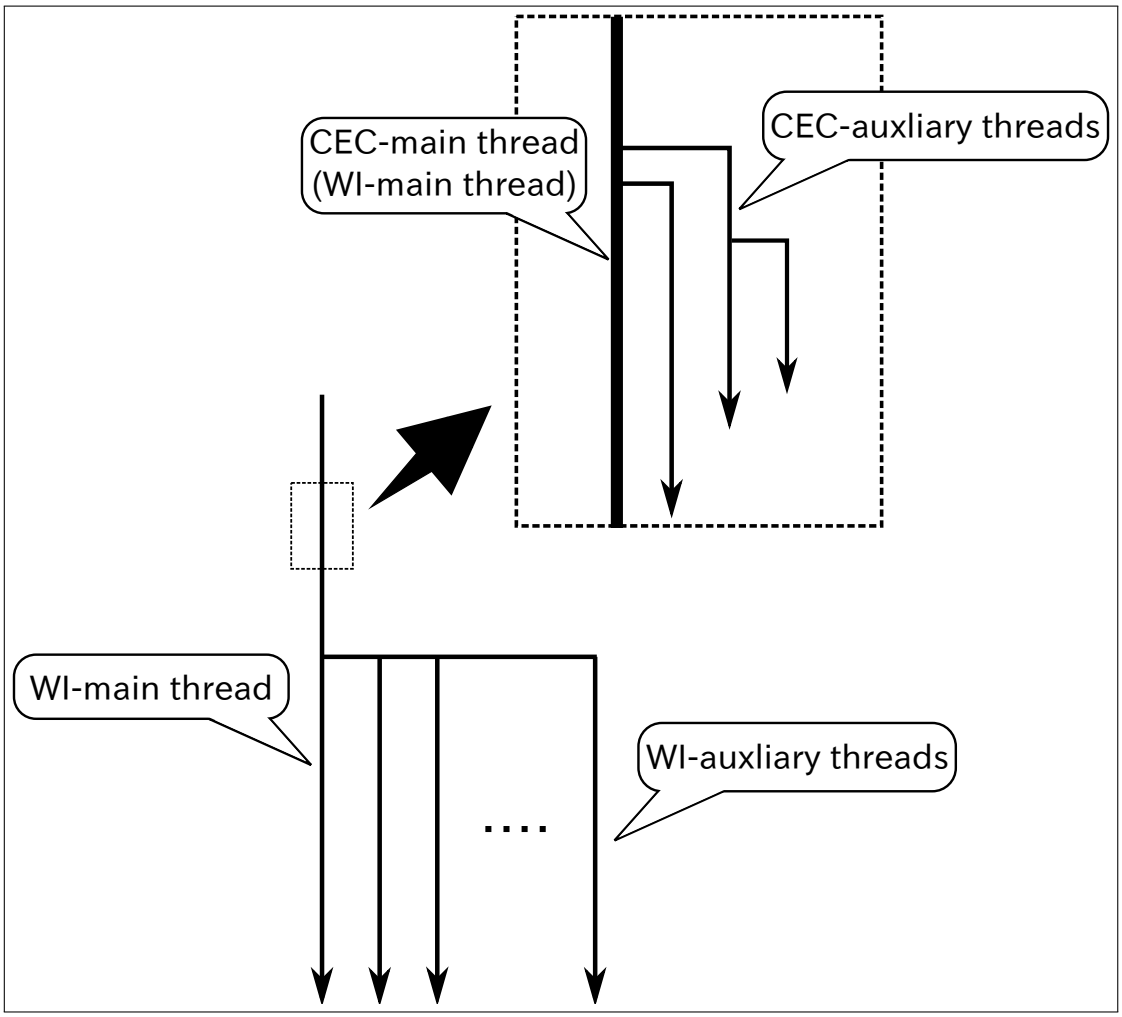

Figure 3.5: WI-main thread, WI-auxiliary thread, CEC-main thread, and CEC-auxiliary thread.

3.4.2.2 Analysis of poly-time simulator-extractor $\mathcal{SE}$.

To prove the statistical CNMZK property, we show that $\mathcal{SE}$ statistically simulates the view of $\mathcal{A}$ and also extracts witnesses for the statements in the right sessions.

Lemma 3.2. *The view of $\mathcal{A}$ simulated by $\mathcal{SE}$ is statistically indistinguishable from the view of $\mathcal{A}$ in the real experiment. Furthermore, except with negligible probability, $\mathcal{SE}$ outputs witnesses for the statements proven by $\mathcal{A}$ in the accepted right sessions that use different identities from the left sessions.*

Proof. In this proof, we use the following claim, which states that the super-polynomial-time simulator-extractor $\hat{\mathcal{SE}}$ statistically simulates the view of $\mathcal{A}$ and also extracts the witnesses from the right sessions.

Claim 3.1. *The view of $\mathcal{A}$ simulated by $\hat{\mathcal{SE}}$ is statistically indistinguishable from the view of $\mathcal{A}$ in the real experiment. Furthermore, except with negligible probability, $\hat{\mathcal{SE}}$*

outputs witnesses for the statements proven by $\mathcal{A}$ in the accepted right sessions that use different identities from the left sessions.

Before proving this claim, we finish the proof of Lemma 3.2. Given Claim 3.1, we can prove Lemma 3.2 by showing that the output of $\mathcal{SE}$ is statistically indistinguishable from that of $\hat{\mathcal{SE}}$. Roughly speaking, this indistinguishability can be shown by observing the following.

- In $\mathcal{SE}$, the emulation of $\hat{\mathcal{SE}}$ is perfect if in every left session that reaches Stage II, the value extracted by the concurrent extractability of CECom is equal to the value that would be extracted by brute force.
- In every such left session, the value extracted by the concurrent extractability of CECom is indeed equal to the value that would be extracted by brute force except with negligible probability. This is because the CECom commitment in Stage I-2 is valid in every such left session except with negligible probability, which in turn is because of the soundness of $\mathsf{WIProof}$ in Stage I-6 and the hiding property of $\mathsf{CCACom}^{1:1}$ in Stage I-3.

We note that there is a subtlety since the concurrent extraction of CECom itself is rewound in $\mathcal{SE}$ when the witnesses are extracted from the right sessions. A formal argument is given below.

Let $\mathsf{CEC\text{-}BAD}$ be the event that during the execution of $\mathcal{SE}$, in a left session that reaches Stage II, the value extracted from the CECom commitment in Stage I-2 is different from the value that would be extracted by the brute-force extraction. Let ϵ be the probability that $\mathsf{CEC\text{-}BAD}$ occurs during the emulation of the WI-main thread (i.e., during the emulation of the execution of $\hat{\mathcal{S}}$). From the concurrent extractability of CECom, if the CECom commitment in Stage I-2 is valid except with negligible probability, $\mathsf{CEC\text{-}BAD}$ occurs only with negligible probability. Hence, we obtain $\epsilon = \mathsf{negl}(n)$ from the following claim.

Claim 3.2. *In $\hat{\mathcal{S}}$, the following holds except with negligible probability: In every left session that reaches Stage II, the CECom commitment in Stage I-2 of this session is valid and its committed value is a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1.*

The proof of Claim 3.2 is given after this proof.

To show the indistinguishability between the output of $\hat{\mathcal{SE}}$ and that of $\mathcal{SE}$, we consider the following hybrid simulator-extractor $\hat{\mathcal{SE}}'$ and $\mathcal{SE}'$.

- $\hat{\mathcal{SE}}'$ (resp., $\mathcal{SE}'$) is the same as $\hat{\mathcal{SE}}$ (resp., $\mathcal{SE}$) except that for each $i \in [m]$, it terminates with output $\mathsf{time\text{-}out}$ if it does not obtain another accepting transcript of the i-th right session after rewinding the WI-main thread (resp., the emulation of the WI-main thread) $1/\epsilon^{1/4}$ times.

First, we show that the output of $\hat{\mathcal{SE}}$ and that of $\hat{\mathcal{SE}}'$ are statistically indistinguishable. From the definition of $\hat{\mathcal{SE}}'$, this indistinguishability holds if $\hat{\mathcal{SE}}'$ outputs $\mathsf{time\text{-}out}$ with at most negligible probability. Thus, to show this indistinguishability, it suffices to show that the probability that $\hat{\mathcal{SE}}$ rewinds WI-main thread more than $1/\epsilon^{1/4}$ times during

the witness extraction of a right session is negligible. To show that this probability is negligible, we do the following. For each $i \in [m]$, let T_i be the random variable representing the number of rewinding during the witness extraction of the i-th right session in $\hat{\mathcal{SE}}$. From a standard "$p \times 1/p$" argument, we can show that we have $\mathrm{E}[T_i] = 1$ for every $i \in [m]$.[16] Thus, from Marcov's inequality, we have

$$\Pr\left[T_i > 1/\epsilon^{1/4}\right] \leq \epsilon^{1/4} = \mathsf{negl}(n)$$

for every $i \in [m]$. Thus, from union bound, we have

$$\Pr\left[\exists i \in [m] \text{ s.t. } T_i > 1/\epsilon^{1/4}\right] \leq m \cdot \mathsf{negl}(n) = \mathsf{negl}(n) \ .$$

As noted above, this implies that the output of $\hat{\mathcal{SE}}$ and that of $\hat{\mathcal{SE}}'$ are statistically indistinguishable.

Next, we show that the output of $\hat{\mathcal{SE}}'$ and that of $\mathcal{SE}'$ are statistically indistinguishable. Since the only difference between $\hat{\mathcal{SE}}'$ and $\mathcal{SE}'$ is how the committed values are extracted from CECom, this indistinguishability holds if CEC-BAD occurs in $\mathcal{SE}'$ with at most negligible probability. For $\ell \in \mathbb{N}$, let ST_ℓ be the random variable representing the internal state of $\mathcal{SE}'$ at the time that $\mathcal{A}$ has sent the ℓ-th messages on the CEC-main thread during the emulation of WI-main thread. Let $\mathsf{CEC\text{-}BAD}_{\mathrm{main}}$ be the event that CEC-BAD occurs during the emulation of the WI-main thread. We say that an internal state st of $\mathcal{SE}'$ is *good w.r.t.* ℓ if we have $\Pr[\mathsf{CEC\text{-}BAD}_{\mathrm{main}} \mid ST_\ell = \mathsf{st}] \leq \epsilon^{1/2}$. Let GOOD_ℓ be the event that $ST_\ell = \mathsf{st}$ holds for an internal state st that is good w.r.t. ℓ. Then, for any ℓ, we have

$$\Pr[\mathsf{CEC\text{-}BAD}_{\mathrm{main}}] \geq \Pr[\mathsf{CEC\text{-}BAD}_{\mathrm{main}} \mid \neg\mathsf{GOOD}_\ell] \Pr[\neg\mathsf{GOOD}_\ell] \geq \epsilon^{1/2} \cdot \Pr[\neg\mathsf{GOOD}_\ell] \ .$$

Then, since we have $\Pr[\mathsf{CEC\text{-}BAD}_{\mathrm{main}}] = \epsilon$ from the definition of ϵ, we have

$$\Pr[\neg\mathsf{GOOD}_\ell] \leq \epsilon^{1/2} = \mathsf{negl}(n)$$

for every ℓ. Thus, from union bound, we have

$$\Pr\left[\bigvee_\ell \neg\mathsf{GOOD}_\ell\right] \leq \mathsf{negl}(n) \ . \tag{3.1}$$

Let GOOD be the event that GOOD_ℓ occurs for every ℓ. Then, from Equation (3.1), we have $\Pr[\mathsf{GOOD}] \geq 1 - \mathsf{negl}(n)$. For $i \in [m]$, let $\mathsf{CEC\text{-}BAD}_i$ be the event that CEC-BAD occurs during the witness extraction of the i-th right session. Then, since the emulation of each WI-auxiliary thread proceeds identically with that of the WI-main thread, and since for each $i \in [m]$ there are at most $1/\epsilon^{1/4}$ WI-auxiliary threads during the witness extraction of the i-th right session, we have

$$\Pr[\mathsf{CEC\text{-}BAD}_i \mid \mathsf{GOOD}] \leq \frac{1}{\epsilon^{1/4}} \cdot \epsilon^{1/2} = \epsilon^{1/4} \ .$$

[16] For any prefix ρ of the transcript immediately before the challenge message of sWIAOK of the i-th right session, let p be the probability that the i-th right session is accepted when the prefix of the transcript is ρ. Then, we have $\mathrm{E}\left[T_i \mid \mathsf{prefix}_\rho\right] = p \cdot 1/p = 1$, where prefix_ρ is the event that the prefix of the transcript is ρ. Thus, we have $\mathrm{E}[T_i] = \sum_\rho \mathrm{E}\left[T_i \mid \mathsf{prefix}_\rho\right] \Pr\left[\mathsf{prefix}_\rho\right] = 1$.

Thus, we have

$$\begin{aligned}
\Pr[\mathsf{CEC\text{-}BAD}] &= \Pr[\mathsf{CEC\text{-}BAD}_{\mathrm{main}}] + \sum_{i=1}^{m} \Pr[\mathsf{CEC\text{-}BAD}_i] \\
&\leq \Pr[\mathsf{CEC\text{-}BAD}_{\mathrm{main}}] + \sum_{i=1}^{m} (\Pr[\neg\mathsf{GOOD}] + \Pr[\mathsf{CEC\text{-}BAD}_i \mid \mathsf{GOOD}]) \\
&\leq \epsilon + \sum_{i=1}^{m} (\mathsf{negl}(n) + \epsilon^{1/4}) \\
&= \mathsf{negl}(n) \ .
\end{aligned}$$

As noted above, this implies that the output of $\hat{\mathcal{SE}}'$ and that of $\mathcal{SE}'$ are statistically indistinguishable.

Finally, we show that the output of $\mathcal{SE}'$ and that of $\mathcal{SE}$ are statistically indistinguishable. Since the output of $\hat{\mathcal{SE}}'$ and that of $\mathcal{SE}'$ are statistically indistinguishable and since $\hat{\mathcal{SE}}'$ outputs time-out with at most negligible probability, $\mathcal{SE}'$ outputs time-out with at most negligible probability. Then, since $\mathcal{SE}'$ is identical to $\mathcal{SE}$ unless $\mathcal{SE}'$ outputs time-out, the output of $\mathcal{SE}'$ and that of $\mathcal{SE}$ are statistically indistinguishable. □

Proof of Claim 3.2. Recall that Claim 3.2 states that during the execution of $\hat{\mathcal{S}}$, in every left session that reaches Stage II, the CECom commitment in Stage I-2 is valid and its committed value is a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1.

Let us say that a left session is *bad* if it reaches Stage II and either the CECom commitment in Stage I-2 is invalid or its committed value is not a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1; a left session is *good* if it is not bad. What we need to prove is that every left session is good except with negligible probability.

Roughly speaking, the proof proceeds as follows. From the soundness of WIProof, if a left session is bad, then in Stage I-4 of this left session, the committed value of the CECom commitment is r_P, which is the committed value of the $\mathsf{CCACom}^{1:1}$ commitment in Stage I-3; thus, before r_P is decommitted to in Stage I-5, we can obtain r_P by extracting the committed value from CECom in Stage I-4. This itself does not contradict the hiding property of $\mathsf{CCACom}^{1:1}$ since $\hat{\mathcal{S}}$ runs in super-polynomial time in the brute-force extraction of CECom. Thus, we consider a hybrid simulator in which the brute-force extraction of CECom is replaced with the concurrent extraction of CECom. Here, since we want to use the hiding property of $\mathsf{CCACom}^{1:1}$, we use the robust concurrent extraction of CECom so that the $\mathsf{CCACom}^{1:1}$ commitment in a left session is not rewound. For details, see below.

Assume for contradiction that there exists $i \in [m]$ such that the i-th left session is bad with non-negligible probability. (Here, the indices of the left sessions are defined by the order in which Stage I-5 begins; the reason why we define the indices in this way will become clear later.) Then, there exists $i^* \in [m]$ such that the first $(i^* - 1)$ left sessions are good except with negligible probability but the i^*-th left session is bad with non-negligible probability. Note that from the soundness of WIProof, when the i^*-th left session is bad, then the CECom commitment in Stage I-4 of the i^*-th left session is valid and its committed value is r_P except with negligible probability, where r_P is the

value committed to in Stage I-3 of the i^*-th left session. In the following, we use BAD to denote the event that the i^*-th left session is bad, and use CHEAT to denote the event that the committed value of the CECom commitment in Stage I-4 is r_P in the i^*-th left session. Then, let us consider the following hybrids.

Hybrid $\hat{\mathcal{S}}_0$ is the same as $\hat{\mathcal{S}}$. From our assumption, BAD occurs in $\hat{\mathcal{S}}_0$ with non-negligible probability. Thus, from the above argument, CHEAT occurs in $\hat{\mathcal{S}}_0$ with non-negligible probability.

Hybrid $\hat{\mathcal{S}}_1$ is the same as $\hat{\mathcal{S}}_0$ except that $\hat{\mathcal{S}}_1$ terminates just before Stage I-5 of the i^*-th left session begins. Clearly, CHEAT still occurs in $\hat{\mathcal{S}}_1$ with non-negligible probability.

Hybrid $\mathcal{S}_1$ emulates $\hat{\mathcal{S}}_1$ in polynomial time as follows.

- At the beginning, a random left session s is chosen. (Here, we guess that session s will be the i^*-th left session.)
- In every left session, in Stage I-2, the committed value (r_V, d) is extracted by the robust concurrent extractor of CECom in such a way that the $\mathsf{CCACom}^{1:1}$ commitment of session s is not rewound. In addition, in the left session s, the committed value is also extracted from the CECom commitment in Stage I-4.

Note that in every left session in which Stage II is executed, the CECom commitment in Stage I-2 is valid except with negligible probability (since such a session is one of the first $(i^* - 1)$ left sessions and therefore it is good except with negligible probability). Thus, the values extracted from the concurrent extractor are equal to the values that would be extracted by the brute-force extraction except with negligible probability; therefore, $\mathcal{S}_1$ statistically emulates $\hat{\mathcal{S}}_1$, and CHEAT occurs in $\mathcal{S}_1$ with non-negligible probability.

Note that session s is the i^*-th left session with non-negligible probability. Then, since CHEAT occurs in $\mathcal{S}_1$ with non-negligible probability, the value extracted from the CECom commitment in Stage I-4 of session s is r_P with non-negligible probability, where r_P is the value committed to in Stage I-3 of session s. Then, since the $\mathsf{CCACom}^{1:1}$ commitment in Stage I-3 of session s is not rewound in $\mathcal{S}_1$, we can break the hiding property of $\mathsf{CCACom}^{1:1}$. Thus, we reach a contradiction. □

3.4.2.3 Analysis of super-poly-time simulator-extractor $\hat{\mathcal{SE}}$.

It remains to prove Claim 3.1, which states that (i) super-polynomial-time simulator-extractor $\hat{\mathcal{SE}}$ statistically simulates the real view of $\mathcal{A}$ and (ii) $\hat{\mathcal{SE}}$ also extracts a valid witness from every accepted right session in the simulated view.

Proof of Claim 3.1. First, we observe that the output of $\hat{\mathcal{S}}$ is statistically indistinguishable from the real view of $\mathcal{A}$. Since $\hat{\mathcal{SE}}$ simulates the view of $\mathcal{A}$ by executing $\hat{\mathcal{S}}$, this implies that $\hat{\mathcal{SE}}$ statistically simulates the real view of $\mathcal{A}$. Recall that in $\hat{\mathcal{S}}$, each left session is simulated by extracting (r_V, d) from the CECom commitment in Stage I-2 and

giving a sWIAOK proof in Stage II with witness (r_V, d). From Claim 3.2 (which states that the CECom commitment in Stage I-2 is a valid commitment to a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1 in every session that reaches Stage II), the value (r_V, d) that is extracted from the CECom commitment in Stage I-2 is a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment of Stage I-1 in each left session that reaches Stage II. Thus, from the statistical witness indistinguishability of sWIAOK, the output of $\hat{\mathcal{S}}$ is statistically indistinguishable from the real view of $\mathcal{A}$.

Next, we show that $\hat{\mathcal{SE}}$ outputs $\mathsf{fail}_{\mathsf{WI}}$ with at most negligible probability. Since $\hat{\mathcal{SE}}$ outputs $\mathsf{fail}_{\mathsf{WI}}$ when it fails to extract a witness in an accepted right session, this implies that $\hat{\mathcal{SE}}$ extracts a valid witness from every accepted right session except with negligible probability. Assume for contradiction that there exists $\widetilde{i^*} \in [m]$ such that $\hat{\mathcal{SE}}$ outputs $\mathsf{fail}_{\mathsf{WI}}$ during the witness extraction of the $\widetilde{i^*}$-th right session with non-negligible probability. Then, let us consider the following hybrid simulator-extractor $\hat{\mathcal{SE}}_{\widetilde{i^*}}$.

- $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ is the same as $\hat{\mathcal{SE}}$ except that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ tries to extract a witness only from the $\widetilde{i^*}$-th right session (and therefore rewinds the WI-main thread only from the challenge message of sWIAOK of the $\widetilde{i^*}$-th right session).

Clearly, $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ outputs $\mathsf{fail}_{\mathsf{WI}}$ with non-negligible probability. Then, we reach a contradiction roughly as follows.

Step 1. First, we show that in $\hat{\mathcal{SE}}_{\widetilde{i^*}}$, the probability that $\widetilde{r_V}$ is extracted as a witness during the witness extraction of the $\widetilde{i^*}$-th right session is non-negligible, where $\widetilde{r_V}$ is the value chosen by the verifier in Stage I-1 of the $\widetilde{i^*}$-th right session.

Step 2. Next, we define a sequence of hybrid simulator-extractors, where the first hybrid is the same as $\hat{\mathcal{SE}}_{\widetilde{i^*}}$, and we gradually modify the $\widetilde{i^*}$-th right session so that it is independent of $\widetilde{r_V}$ in the last hybrid.

Step 3. Finally, we show that even in the last hybrid, the probability that $\widetilde{r_V}$ is extracted during the witness extraction of the $\widetilde{i^*}$-th right session is non-negligible. Since the $\widetilde{i^*}$-th right session is independent of $\widetilde{r_V}$ in the last hybrid, we reach a contradiction.

Details are given below.

Step 1. Prove that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ extracts $\widetilde{r_V}$. We first prove the following claim.

Claim 3.3. *Let $\widetilde{r_V}$ be the value chosen by the verifier in Stage I-1 of the $\widetilde{i^*}$-th right session. If $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ outputs $\mathsf{fail}_{\mathsf{WI}}$ with non-negligible probability, then in $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ the probability that $\widetilde{r_V}$ is extracted during the witness extraction of the $\widetilde{i^*}$-th right session is non-negligible.*

Proof. Assume for contradiction that $\widetilde{r_V}$ is extracted during the witness extraction of the $\widetilde{i^*}$-th right session with at most negligible probability. Then, since we assume that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ outputs $\mathsf{fail}_{\mathsf{WI}}$ with non-negligible probability, the following occurs in $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ with non-negligible probability:

- $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ obtains two accepting transcript of the $\widetilde{i^*}$-th right session (and therefore that of sWIAOK) such that the commit-messages of sWIAOK are the same,[17] but

- from these two transcript, $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ fails to extract any witness from sWIAOK (i.e., a witness for $\widetilde{x}_{\widetilde{i^*}} \in L$ or a valid decommitment of the Stage I-1 commitment).

We first show that when the above occurs, the two accepting sWIAOK transcripts are admissible except with negligible probability. (Recall that a pair of accepted transcripts of sWIAOK are admissible if their commit-messages are the same but their challenge-messages are different.) Toward this end, it suffices to show that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ chooses the same challenge-message of sWIAOK on the wi-main thread and a wi-auxiliary thread with at most negligible probability. This can be shown as follows.

- From a standard argument, we can show that the expected number of rewinding of the wi-main thread is 1 in $\hat{\mathcal{SE}}_{\widetilde{i^*}}$.[18] Thus, the probability that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ rewinds the wi-main thread more than $2^{n/2}$ times is at most $2^{-n/2}$. Furthermore, under the condition that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ rewinds the wi-main thread at most $2^{n/2}$ times, the probability that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ chooses the same challenge-message on the wi-main thread and a wi-auxiliary thread is at most $2^{n/2} \cdot 2^{-n} = 2^{-n/2}$. Thus, the probability that $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ chooses the same challenge-message on the wi-main thread and a wi-auxiliary thread is at most $2^{-n/2} + 2^{-n/2} = \mathsf{negl}(n)$.

Thus, with non-negligible probability $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ obtains two admissible transcripts of sWIAOK from which no witness can be computed.

We then reach a contradiction as follows. Since sWIAOK is a parallel version of Blum's Hamiltonian-cycle protocol, if no witness is extracted from two admissible transcripts of sWIAOK, a $\mathsf{Com}_{\mathsf{SH}}$ commitment in the commit-messages of those transcripts is decommitted to two different values. Thus, we derive a contradiction by breaking the binding property of $\mathsf{Com}_{\mathsf{SH}}$ using $\hat{\mathcal{SE}}_{\widetilde{i^*}}$. A problem is that since $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ runs in super-polynomial time, the *computational* biding property of $\mathsf{Com}_{\mathsf{SH}}$ may not hold in $\hat{\mathcal{SE}}_{\widetilde{i^*}}$. To overcome this problem, we consider hybrid simulator-extractor $\mathcal{SE}_{\widetilde{i^*}}$ that emulates the execution of $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ in polynomial time. Specifically, $\mathcal{SE}_{\widetilde{i^*}}$ emulates $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ in the same way as $\mathcal{SE}$ emulates $\hat{\mathcal{SE}}$ (i.e., by using the concurrent extractability of CECom instead of the brute-force extraction) except for the following.

- During the emulation of the wi-main thread, the value (r_V, d) is extracted in Stage I-2 of each left session by using the *robust* concurrent extractability of CECom so that the commit-message of sWIAOK in the $\widetilde{i^*}$-th right session is not rewound.

As in the proof of Lemma 3.2, we can show that $\mathcal{SE}_{\widetilde{i^*}}$ statistically emulates the execution of $\hat{\mathcal{SE}}_{\widetilde{i^*}}$. Thus, with non-negligible probability, $\mathcal{SE}_{\widetilde{i^*}}$ obtains two valid decommitments of a $\mathsf{Com}_{\mathsf{SH}}$ commitment (in the commit-messages of sWIAOK of the $\widetilde{i^*}$-th right session) such that decommitted values are different. Then, since $\mathcal{SE}_{\widetilde{i^*}}$ runs in polynomial time and since the commit-messages of sWIAOK (and therefore the $\mathsf{Com}_{\mathsf{SH}}$ commitment) of

[17] Recall that $\mathsf{WIProof}$ consists of three stages: commit, challenge, and response.
[18] See Footnote 16.

the $\widetilde{i^*}$-th right session is not rewound in $\mathcal{SE}_{\widetilde{i^*}}$,[19] we can break the binding property of $\mathsf{Com}_{\mathsf{SH}}$. Thus, we reach a contradiction. □

Step 2. Introduce hybrid simulator-extractor. Next, we introduce hybrid simulator-extractors. To clarify the exposition, we first define a sequence of hybrid simulators by gradually modifying $\hat{\mathcal{S}}$ and then define the hybrid simulator-extractors by using them. Below, when we refer to a particular stage of sCNMZK, we always means the corresponding stage of sCNMZK in the $\widetilde{i^*}$-th right session.

Hybrid simulator h-$\hat{\mathcal{S}}_0$ is identical with $\hat{\mathcal{S}}$.

Hybrid simulator h-$\hat{\mathcal{S}}_1$ is the same as h-$\hat{\mathcal{S}}_0$ except that $\widetilde{r}_P$ is extracted by brute force in Stage I-3 and the committed value of the CECom commitment in Stage I-4 is switched from 0^n to $\widetilde{r}_P$.

Hybrid simulator h-$\hat{\mathcal{S}}_2$ is the same as h-$\hat{\mathcal{S}}_1$ except that in Stage I-6, the WIProof proof is computed by using a witness for the fact that the committed value of the CECom commitment in Stage I-4 is $\widetilde{r}_P$.

Hybrid simulator h-$\hat{\mathcal{S}}_3$ is the same as h-$\hat{\mathcal{S}}_2$ except that in Stage I-2, the committed value of the CECom commitment is switched from $(\widetilde{r}_V, \widetilde{d})$ to $(0^{|\widetilde{r}_V|}, 0^{|\widetilde{d}|})$.

Hybrid simulator h-$\hat{\mathcal{S}}_4$ is the same as h-$\hat{\mathcal{S}}_3$ except that in Stage I-1, the committed value of the $\mathsf{Com}_{\mathsf{SB}}$ commitment is switched from $\widetilde{r}_V$ to 0^n.

Then, for each $k \in \{0, \ldots, 4\}$, hybrid simulator-extractor h-$\hat{\mathcal{SE}}_k$ is defined as follows.

Hybrid simulator-extractor h-$\hat{\mathcal{SE}}_k$ is the same as $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ except that the execution of $\hat{\mathcal{S}}$ is replaced with that of h-$\hat{\mathcal{S}}_k$. The output of h-$\hat{\mathcal{SE}}_k$ is the value extracted during the witness extraction of the $\widetilde{i^*}$-th right session.

Note that the value $\widetilde{r}_V$ is not used anywhere in h-$\hat{\mathcal{SE}}_4$.

Step 3. Prove that $\widetilde{r}_V$ is extracted in every hybrid. Finally, we show that $\widetilde{r}_V$ is extracted with non-negligible probability in each hybrid. First, we consider h-$\hat{\mathcal{SE}}_1$.

Claim 3.4. *Let $\widetilde{r}_V$ be the value chosen by the verifier in Stage I-1 of the $\widetilde{i^*}$-th right session. If $\hat{\mathcal{SE}}_{\widetilde{i^*}}$ outputs $\mathsf{fail}_{\mathsf{WI}}$ with non-negligible probability, then in h-$\hat{\mathcal{SE}}_1$ the probability that $\widetilde{r}_V$ is extracted during the witness extraction of the $\widetilde{i^*}$-th right session is non-negligible.*

Proof. In this proof, we use intermediate hybrid simulator-extractors in which the CECom commitment in Stage I-4 of the $\widetilde{i^*}$-th right session is gradually modified. Again, we first introduce hybrid simulators. Recall that a CECom commitment consists of $\ell = \omega(R_{\mathsf{SH}}(n) \log n)$ ExtCom commitments (cf. Figure 2.2 in Section 2.3.3). Then, the intermediate hybrid simulators h-$\hat{\mathcal{S}}_{0:0}, \ldots, h$-$\hat{\mathcal{S}}_{0:\ell}$ are defined as follows.

[19] Note that the commit-messages of sWIAOK of the $\widetilde{i^*}$-th right session appear only on the WI-main thread.

Hybrid simulator $h\text{-}\hat{\mathcal{S}}_{0:0}$ is the same as $h\text{-}\hat{\mathcal{S}}_0$ except that $\widetilde{r}_P$ is extracted by brute force in Stage I-3 of the $\widetilde{i^*}$-th right session.

Hybrid simulator $h\text{-}\hat{\mathcal{S}}_{0:k}$ ($k \in [\ell]$) is the same as $h\text{-}\hat{\mathcal{S}}_{0:k-1}$ except that the committed value of the k-th ExtCom commitment in the CECom commitment of Stage I-4 is switched from 0^n to $\widetilde{r}_P$ in the $\widetilde{i^*}$-th right session.

Then, for each $k \in \{0, \ldots, \ell\}$, hybrid simulator-extractor $h\text{-}\hat{\mathcal{SE}}_{0:k}$ is defined as follows.

Hybrid simulator-extractor $h\text{-}\hat{\mathcal{SE}}_{0:k}$ is the same as $h\text{-}\hat{\mathcal{SE}}_0$ except that the execution of $h\text{-}\hat{\mathcal{S}}_0$ is replaced with that of $h\text{-}\hat{\mathcal{S}}_{0:k}$.

Note that $h\text{-}\hat{\mathcal{SE}}_{0:\ell}$ is identical with $h\text{-}\hat{\mathcal{SE}}_1$.

Below, we show that for every $k \in [\ell]$, the output of $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and that of $h\text{-}\hat{\mathcal{SE}}_{0:k}$ are indistinguishable. (Recall that the outputs of $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and $h\text{-}\hat{\mathcal{SE}}_{0:k}$ are the value extracted in the $\widetilde{i^*}$-th right session.) Since the probability that $\widetilde{r}_V$ is extracted in $h\text{-}\hat{\mathcal{SE}}_{0:0}$ is non-negligible from Claim 3.3, this suffices to prove Claim 3.4.

Roughly speaking, we show this indistinguishability as follows. Since $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and $h\text{-}\hat{\mathcal{SE}}_{0:k}$ differ only in the committed values of a ExtCom commitment, we use the hiding property of the ExtCom commitment to show the indistinguishability. A problem is that we cannot use it directly since $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and $h\text{-}\hat{\mathcal{SE}}_{0:k}$ run in super-polynomial time. To overcome this problem, we observe that the only super-polynomial computations in $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and $h\text{-}\hat{\mathcal{SE}}_{0:k}$ are the brute-force extraction of $\mathsf{CCACom}^{1:1}$ (in the $\widetilde{i^*}$-th right session) and those of CECom (in the left sessions). Based on this observation, we first show that the execution of $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and $h\text{-}\hat{\mathcal{SE}}_{0:k}$ can be emulated in polynomial-time by using the one-session committed-value oracle O of $\mathsf{CCACom}^{1:1}$ and the concurrent extractability of CECom. We then combine the 4-robustness of $\mathsf{CCACom}^{1:1}$ with the hiding property of ExtCom (which has only four rounds) to argue that the output of $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and that of $h\text{-}\hat{\mathcal{SE}}_{0:k}$ are indistinguishable. To formally implement this idea, we need to make sure that the ExtCom commitment and the $\mathsf{CCACom}^{1:1}$ commitment are not rewound during the concurrent extraction of CECom. Details are given below.

First, we introduce hybrid simulator-extractors $h\text{-}\mathcal{SE}^{O}_{0:k-1}$ and $h\text{-}\mathcal{SE}^{O}_{0:k}$, where O is the one-session committed-value oracle of $\mathsf{CCACom}^{1:1}$. Hybrid $h\text{-}\mathcal{SE}^{O}_{0:k}$ (resp., $h\text{-}\mathcal{SE}^{O}_{0:k-1}$) emulates $h\text{-}\hat{\mathcal{SE}}_{0:k}$ (resp., $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$) in the same way as $\mathcal{SE}$ emulates $\hat{\mathcal{SE}}$ except for the following.

- During the emulation of the WI-main thread, the value (r_V, d) is extracted in Stage I-2 of each left session by using the robust concurrent extractability so that the $\mathsf{CCACom}^{1:1}$ commitment in Stage I-3 and the k-th ExtCom commitment in the CECom commitment of Stage I-4 are not rewound in the $\widetilde{i^*}$-th right session. In addition, in the $\widetilde{i^*}$-th right session, the committed value of $\mathsf{CCACom}^{1:1}$ is extracted by forwarding the commitment to O. Note that the $\mathsf{CCACom}^{1:1}$ commitment in the $\widetilde{i^*}$-th right session is not rewound and therefore it can be forwarded to O.

Next, we show that for each $h \in \{k-1, k\}$, the output of $h\text{-}\hat{\mathcal{SE}}_{0:h}$ and that of $h\text{-}\mathcal{SE}^{O}_{0:h}$ are indistinguishable. This can be proven in a similar way to Lemma 3.2. In particular, we can use the same argument if we use the following claim instead of Claim 3.2.

Claim 3.5. *In h-$\hat{\mathcal{S}}_{0:h}$ for each $h \in \{k-1, k\}$, the following holds except with negligible probability: In every left session that reaches Stage II, the* CECom *commitment in Stage I-2 of this session is valid and its committed value is a valid decommitment of the* $\mathsf{Com}_{\mathsf{SB}}$ *commitment in Stage I-1.*

Claim 3.5 can be proven in a similar way to Claim 3.2. For completeness, we give the proof below. (Many texts are taken verbatim from the proof of Claim 3.2)

Proof of Claim 3.5. Let us say that a left session is *bad* if it reaches Stage II and either the CECom commitment in Stage I-2 is invalid or its committed value is not a valid decommitment of the $\mathsf{Com}_{\mathsf{SB}}$ commitment in Stage I-1; a left session is *good* if it is not bad. What we want to prove is that every left session is good except with negligible probability.

Roughly speaking, the proof proceeds as follows. From the soundness of $\mathsf{WIProof}$, if a left session is bad, then in Stage I-4 of this left session, the committed value of the CECom commitment is r_P, which is the committed value of the $\mathsf{CCACom}^{1:1}$ commitment in Stage I-3; thus, before r_P is decommitted to in Stage I-5, we can obtain r_P by extracting the committed value from CECom in Stage I-4. This itself does not contradict the hiding property of $\mathsf{CCACom}^{1:1}$ since h-$\hat{\mathcal{S}}_{0:h}$ runs in super-polynomial time in the brute-force extraction of CECom and $\mathsf{CCACom}^{1:1}$. Thus, we again replace the brute-force extraction with the concurrent extraction of CECom and an oracle access to the one-session committed-value oracle O of $\mathsf{CCACom}^{1:1}$, and use the one-one CCA-security of $\mathsf{CCACom}^{1:1}$ instead of its hiding property. Here, since we want to use the one-one CCA-security of $\mathsf{CCACom}^{1:1}$, we perform the concurrent extraction of CECom so that the $\mathsf{CCACom}^{1:1}$ commitment in a left session and the $\mathsf{CCACom}^{1:1}$ in the $\widetilde{i^*}$-th right session are not rewound. Details are given below.

Assume for contradiction that there exists $h \in \{k-1, k\}$ such that in h-$\hat{\mathcal{S}}_{0:h}$, a left session is bad with non-negligible probability. (Here, the indices of the left sessions are determined by the order in which Stage I-5 begins; the reason why we define the indices in this way will become clear later.) Then, there exists $i^* \in [m]$ such that in h-$\hat{\mathcal{S}}_{0:h}$, the first (i^*-1) left sessions are good except with negligible probability but the i^*-th left session is bad with non-negligible probability. Note that from the soundness of $\mathsf{WIProof}$, when the i^*-th left session is bad, the committed value of the CECom commitment in Stage I-4 is r_P in the i^*-th left session except with negligible probability, where r_P is the value committed to in Stage I-3 of the i^*-th left session. In the following, we use BAD to denote the event that the i^*-th left session is bad, and use CHEAT to denote the event that the committed value of the CECom commitment in Stage I-4 is r_P in the i^*-th left session. Then, let us consider the following hybrids.

Hybrid simulator h-$\hat{\mathcal{S}}_{0:h:0}$ is the same as h-$\hat{\mathcal{S}}_{0:h}$. From our assumption, BAD occurs in h-$\hat{\mathcal{S}}_{0:h:0}$ with non-negligible probability. Thus, from the above argument, CHEAT occurs in h-$\hat{\mathcal{S}}_{0:h:0}$ with non-negligible probability.

Hybrid simulator h-$\hat{\mathcal{S}}_{0:h:1}$ is the same as h-$\hat{\mathcal{S}}_{0:h:0}$ except that h-$\hat{\mathcal{S}}_{0:h:1}$ terminates just before Stage I-5 of the i^*-th left session begins. Clearly, CHEAT still occurs in h-$\hat{\mathcal{S}}_{0:h:1}$ with non-negligible probability.

Hybrid simulator h-$\mathcal{S}^{O}_{0:h:1}$ emulates h-$\hat{\mathcal{S}}_{0:h:1}$ in polynomial time as follows.

- At the beginning, a random left session s is chosen. (Here, we guess that session s will be the i^*-th left session.)
- In every left session, in Stage I-2, the committed value (r_V, d) is extracted by the robust concurrent extractor of CECom in such a way that the $\mathsf{CCACom}^{1:1}$ commitment of left session s and the $\mathsf{CCACom}^{1:1}$ commitment of the $\widetilde{i^*}$-th right session are not rewound. In addition, in the $\widetilde{i^*}$-th right session, the committed value of $\mathsf{CCACom}^{1:1}$ is extracted by forwarding the commitment to O.
- In left session s, the committed value is also extracted in Stage I-4 by the robust concurrent extractor of CECom without rewinding the $\mathsf{CCACom}^{1:1}$ commitment of the $\widetilde{i^*}$-th right session.

Note that when Stage II of a left session is executed, the CECom commitment in Stage I-2 of that session is valid except with negligible probability (since that session is one of the first $(i^* - 1)$ left sessions and therefore it is good except with negligible probability). Thus, the values extracted from the concurrent extractor are equal to the values that would be extracted by brute force except with negligible probability; therefore, $h\text{-}\mathcal{S}^{O}_{0:h:1}$ statistically emulates $h\text{-}\hat{\mathcal{S}}_{0:h:1}$, and CHEAT occurs in $h\text{-}\mathcal{S}^{O}_{0:h:1}$ with non-negligible probability.

Note that session s is the i^*-th left session with non-negligible probability. Then, since CHEAT occurs in $h\text{-}\mathcal{S}^{O}_{0:h:1}$ with non-negligible probability, r_P is extracted from the CECom commitment in Stage I-4 of session s with non-negligible probability, where r_P is the value committed to in Stage I-3 of session s. Then, since the $\mathsf{CCACom}^{1:1}$ commitment of session s is not rewound in $h\text{-}\mathcal{S}^{O}_{0:h:1}$, we can break the one-one CCA security of $\mathsf{CCACom}^{1:1}$. Thus, we reach a contradiction. □

As argued above, Claim 3.5 implies that for each $h \in \{k-1, k\}$, the outputs of $h\text{-}\hat{\mathcal{SE}}_{0:h}$ and $h\text{-}\mathcal{SE}^{O}_{0:h}$ are indistinguishable.

To show that the outputs of $h\text{-}\hat{\mathcal{SE}}_{0:k-1}$ and $h\text{-}\hat{\mathcal{SE}}_{0:k}$ are indistinguishable, it remains to prove that the outputs of $h\text{-}\mathcal{SE}^{O}_{0:k-1}$ and $h\text{-}\mathcal{SE}^{O}_{0:k}$ are indistinguishable. This can be shown as follows. Observe that $h\text{-}\mathcal{SE}^{O}_{0:k-1}$ and $h\text{-}\mathcal{SE}^{O}_{0:k}$ differ only in the k-th ExtCom commitment of the CECom commitment of the $\widetilde{i^*}$-th right session, and this ExtCom commitment is not rewound in $h\text{-}\mathcal{SE}^{O}_{0:k-1}$ and $h\text{-}\mathcal{SE}^{O}_{0:k}$. In addition, $h\text{-}\mathcal{SE}^{O}_{0:k-1}$ and $h\text{-}\mathcal{SE}^{O}_{0:k}$ run in polynomial time given oracle access to the one-session committed-value oracle O of $\mathsf{CCACom}^{1:1}$. Thus, from the hiding property of ExtCom and the 4-robustness of $\mathsf{CCACom}^{1:1}$, the output of $\mathcal{SE}^{O}_{0:k-1}$ and that of $h\text{-}\mathcal{SE}^{O}_{0:k}$ are indistinguishable.

Thus, we conclude that the probability that $\widetilde{r_V}$ is extracted in $h\text{-}\hat{\mathcal{SE}}_1$ is non-negligible. This concludes the proof of Claim 3.4. □

By using essentially the same argument as in the proof of Claim 3.4, we can show that $\widetilde{r_V}$ is extracted with non-negligible probability also in $h\text{-}\hat{\mathcal{SE}}_2$, $h\text{-}\hat{\mathcal{SE}}_3$, and $h\text{-}\hat{\mathcal{SE}}_4$. For example, let us consider $h\text{-}\hat{\mathcal{SE}}_2$. Recall that $h\text{-}\hat{\mathcal{SE}}_2$ differs from $h\text{-}\hat{\mathcal{SE}}_1$ only in that the different witness is used in $\mathsf{WIProof}$ of the $\widetilde{i^*}$-th right session. Then, in the same way as in the proof of Claim 3.4, we can define hybrid simulator-extractors $h\text{-}\mathcal{SE}^{O}_1$ and $h\text{-}\mathcal{SE}^{O}_2$ such that the following hold.

- Given oracle access to the one-session committed-value oracle O of $\mathsf{CCACom}^{1:1}$, both $h\text{-}\mathcal{SE}_1^O$ and $h\text{-}\mathcal{SE}_2^O$ run in polynomial-time.
- For each $k \in \{1,2\}$, the probability that $\widetilde{r}_V$ is extracted in $h\text{-}\mathcal{SE}_k^O$ is statistically close to the probability in $h\text{-}\hat{\mathcal{SE}}_k$.
- $h\text{-}\mathcal{SE}_1^O$ and $h\text{-}\mathcal{SE}_2^O$ differ only in the witness used in $\mathsf{WIProof}$ of the $\widetilde{i^*}$-th right session, and this $\mathsf{WIProof}$ is not rewound in both $h\text{-}\mathcal{SE}_1^O$ and $h\text{-}\mathcal{SE}_2^O$.

Assume for contradiction that $\widetilde{r}_V$ is extracted in $h\text{-}\hat{\mathcal{SE}}_2$ only with negligible probability. Then, since $\widetilde{r}_V$ is extracted in $h\text{-}\hat{\mathcal{SE}}_1$ with non-negligible probability, we can break witness indistinguishability of $\mathsf{WIProof}$ and the 4-robustness of $\mathsf{CCACom}^{1:1}$ by using $h\text{-}\mathcal{SE}_1^O$ and $h\text{-}\mathcal{SE}_2^O$. Thus, $\widetilde{r}_V$ is extracted in $h\text{-}\hat{\mathcal{SE}}_2$ with non-negligible probability. In this way, we can show that $\widetilde{r}_V$ is extracted also in $h\text{-}\hat{\mathcal{SE}}_3$ and $h\text{-}\hat{\mathcal{SE}}_4$ with non-negligible probability.

Concluding the proof of Claim 3.1. In $h\text{-}\hat{\mathcal{SE}}_4$, the $\widetilde{i^*}$-th right session is independent of $\widetilde{r}_V$, and therefore the probability that $\widetilde{r}_V$ is extracted is negligible. However, we show above that this probability is non-negligible. Thus, we reach a contradiction. □

This concludes the proof of Theorem 3.2. □

3.5 Appendices to Chapter 3

3.5.1 Constant-round One-one CCA-secure Commitment Scheme from OWF

In this section, we observe that from a result by Goyal et al. [GLP+15], it follows almost immediately that we can obtain a constant-round one-one CCA-secure commitment scheme from one-way functions.

Theorem 3.3. *Assume the existence of one-way functions. Then, for any constant $\kappa \in \mathbb{N}$, there exists a constant-round κ-robust one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$.*

We use the following building blocks, all of which can be constructed from one-way functions.

- Constant-round commitment scheme NMCom that is non-malleable w.r.t. itself and any 4-round protocol. Specifically, we use the scheme by Lin and Pass [LP11b, LP15]. We remark that, like many other non-malleable commitment schemes, the scheme by [LP11b, LP15] also satisfies extractability.[20]

[20] In the scheme of [LP11b, LP15], the committer proves by a witness-indistinguishable proof of knowledge system that it knows either the committed value or trapdoor information. Since the scheme is designed so that the trapdoor is hidden from the committer, the committed value can be extracted by extracting the witness from the witness-indistinguishable proof.

- Four-round witness-indistinguishable proof $\mathsf{WIProof}$ (see Section 3.3.3).

- Constant-round zero-knowledge argument ZKArg [GK96a].

- Concurrently extractable commitment scheme CECom of Micciancio et al. [MOSV06] with parameter $\ell = \max(\kappa, r_{\text{NM}}, 4) + 1$, where r_{NM} is the round complexity of NMCom. (see Section 3.3.1).

  When $\ell = \max(\kappa, r_{\text{NM}}, 4) + 1 = O(1)$, CECom does not guarantee concurrent extractability. It is easy to see, however, that it guarantees the following "robust extractability" property: For any adversarial committer C^* that commits to a value in a *single* session of CECom and simultaneously participates an arbitrary $\max(\kappa, r_{\text{NM}}, 4)$-round protocol Π, the extractor can extract the value that is committed by C^* without rewinding Π. For details, see Section 3.5.2.

$\mathsf{CCACom}^{1:1}$ is shown in Figure 3.6. We remark that $\mathsf{CCACom}^{1:1}$ is almost identical to the CCA-secure commitment scheme of Goyal et al. [GLP+15]; essentially, the only difference is the parameter ℓ of CECom. We prove its one-one CCA security in Section 3.5.1.1 and prove its robustness in Section 3.5.1.2.

Below, CECom is the scheme of Micciancio et al. [MOSV06] with parameter $\ell = \max(\kappa, r_{\text{NM}}, 4) + 1$, where r_{NM} is the round complexity of NMCom.

Commit Phase

To commit to $v \in \{0,1\}^n$, the committer C does the following with the receiver R.

Stage 1. R chooses random $r \in \{0,1\}^n$ and commits to r by using CECom. R then proves the validity of this CECom commitment by using ZKArg.

Stage 2. C commits to v by using CECom.

Stage 3. C commits to 0^n by using NMCom.

Stage 4. R decommits the CECom commitment in Stage 1 to r.

Stage 5. C proves the following by using $\mathsf{WIProof}$:

- the CECom commitment in Stage 2 is valid, or
- the committed value of the NMCom commitment in Stage 3 is r.

Decommit Phase

C decommits the CECom commitment in Stage 2 to v.

Figure 3.6: Constant-round one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$.

3.5.1.1 Proof of One-one CCA Security

For any adversary $\mathcal{A}$ that interacts with the committed-value oracle only in a single session, we show that the following indistinguishability holds.

$$\{\mathsf{IND}_0(\langle C, R\rangle, \mathcal{A}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \stackrel{c}{\approx} \{\mathsf{IND}_1(\langle C, R\rangle, \mathcal{A}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \quad .$$

Toward this end, we consider a sequence of hybrid experiments in which the left session of $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{A}, n, z)$ is gradually modified so that $\mathcal{A}$ receives no information about v_b in the last hybrid.

Hybrid $H_0^b(n, z)$ is the same as $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{A}, n, z)$.

Hybrid $H_1^b(n, z)$ is the same as $H_0^b(n, z)$ except for the following.

- In Stage 1 of the left session, the committed value r of the CECom commitment is extracted by brute force. If the CECom commitment is invalid or has more then one committed value, r is defined to be a random value.
- In Stage 3 of the left session, the committed value of NMCom is switched from 0^n to r.

Hybrid $H_2^b(n, z)$ is the same as $H_1^b(n, z)$ except that in Stage 5 of the left session, the WIProof proof is computed by using the witness for the fact that the committed value of the NMCom commitment in Stage 3 is r. (Notice that from the statistical binding property of CECom, the probability that $\mathcal{A}$ correctly decommits the CECom commitment in Stage 1 to a value other than r is negligible.)

Hybrid $H_3^b(n, z)$ is the same as $H_2^b(n, z)$ except that in Stage 2 of the left session, the committed value of CECom is switched from v_b to 0^n.

For each $i \in \{0, 1, 2, 3\}$ and $b \in \{0, 1\}$, let $\mathrm{HYB}_i^b(n, z)$ be the random variable representing the output of $H_i^b(n, z)$. From the construction, $\mathcal{A}$ receives no information about v_b in $H_3^0(n, z)$ and $H_3^1(n, z)$ and hence $\mathrm{HYB}_3^0(n, z)$ and $\mathrm{HYB}_3^1(n, z)$ are identically distributed. Therefore, to show the indistinguishability between the above two ensembles, it suffices to prove that the outputs of each neighboring hybrids are computationally indistinguishable.

Our strategy for proving the indistinguishability of each neighboring hybrids is to reduce their indistinguishability to the security of NMCom, WIProof, and CECom. The problem of this strategy is the existence of the committed-value oracle: Since the oracle runs in super-polynomial time, the security of NMCom, WIProof, and CECom might not hold against the adversaries that interact with the oracle. We overcome this problem by showing that the oracle can be emulated efficiently without "disturbing" the security of NMCom, WIProof, and CECom. Specifically, we show that the oracle can be emulated by extracting the committed value of the CECom commitment in Stage 2 of the right session using the extractability of CECom; since CECom provides a *robust* extractability property, the extraction from CECom does not disturb the security of NMCom, WIProof, and CECom. We remark that in the formal argument given below, we first show that $\mathcal{A}$ "cheats" in the hybrids only with negligible probability, meaning

that in the right session, the committed value of the NMCom commitment in Stage 3 is equal to the committed value of the CECom commitment in Stage 1 only with negligible probability. Showing that $\mathcal{A}$ cheats only with negligible probability is crucial to showing that the oracle can be efficiently emulated. In particular, once we show that $\mathcal{A}$ cheats only with negligible probability, we can use the soundness of WIProof to argue that the CECom commitment in Stage 2 is valid in the accepted right session except with negligible probability, and thus we can conclude that the extracted value is equal to the committed value when the right session is accepted. The formal argument is given below.

Let us say that $\mathcal{A}$ *cheats* if the committed value of NMCom in Stage 3 is equal to the committed value $\widetilde{r}$ of CECom in Stage 1 in the accepted right session. First, we show that $\mathcal{A}$ cheats in $H_0^b(n,z)$ only with negligible probability.

Claim 3.6. *The probability that $\mathcal{A}$ cheats in $H_0^b(n,z)$ is negligible for each $b \in \{0,1\}$.*

Proof. Roughly speaking, this claim follows from the hiding property of CECom—when the adversary cheats, we can obtain $\widetilde{r}$ by extracting the committed value from NMCom, and thus we can obtain the committed value of a CECom commitment before it is decommitted to. To formally implement this idea, it is important that no super-polynomial-time computation is performed during the execution of CECom in Stage 1 of the right session. Fortunately, in $H_0^b(n,z)$ no super-polynomial-time computation is indeed performed during CECom of the right session, as super-polynomial-time computation is performed only at the end of the right session. (Recall the in the setting of one-one CCA security, $\mathcal{A}$ interacts with the oracle only in a single session.) The formal argument is given below.

Assume for contradiction that there exists $b \in \{0,1\}$ such that $\mathcal{A}$ cheats in $H_0^b(n,z)$ with non-negligible probability. Fix any such b. To derive a contradiction, we consider the following hybrid experiments.

Hybrid $H_{0:1}^b(n,z)$ is the same as $H_0^b(n,z)$ except that in Stage 3 of the right session, the committed value of the NMCom commitment is extracted by using the extractability of NMCom.

Clearly, the probability that $\mathcal{A}$ cheats is still non-negligible in $H_{0:1}^b(n,z)$. Hence, from the extractability of NMCom, the extracted value is equal to $\widetilde{r}$ with non-negligible probability.

Hybrid $H_{0:2}^b(n,z)$ is the same as $H_{0:1}^b(n,z)$ except that in Stage 1 of the right session, the ZKArg proof is generated by using the simulator of ZKArg.

From the zero-knowledge property of ZKArg, the probability that $\widetilde{r}$ is extracted from NMCom is still non-negligible in $H_{0:2}^b(n,z)$.

We derive a contradiction by constructing an adversary $\mathcal{B}$ that breaks the hiding property of CECom. Externally, $\mathcal{B}$ interacts with a committer of CECom: It sends random $\widetilde{r}_0, \widetilde{r}_1 \in \{0,1\}^n$ to the committer and receives a CECom commitment in which either $\widetilde{r}_0$ or $\widetilde{r}_1$ is committed. Internally, $\mathcal{B}$ invokes $\mathcal{A}$ and emulates $H_{0:2}^b(n,z)$ for $\mathcal{A}$ honestly except that in Stage 1 of the right session, $\mathcal{B}$ forwards the CECom commitment from the external committer to internal $\mathcal{A}$. Finally, if the value extracted from NMCom is

$\widetilde{r}_1$ in internally emulated $H^b_{0:2}(n,z)$, $\mathcal{B}$ outputs 1, and otherwise, it outputs 0. If $\mathcal{B}$ receives a commitment to $\widetilde{r}_1$, it outputs 1 with non-negligible probability from the above argument. On the other hand, if $\mathcal{B}$ receives a commitment to $\widetilde{r}_0$, it outputs 1 only with negligible probability since internal $\mathcal{A}$ receives no information about $\widetilde{r}_1$. Hence, $\mathcal{B}$ breaks the hiding property of CECom. □

Next, we show that $\mathcal{A}$ cheats only with negligible probability in $H^b_1(n,z)$, and we use it to prove that $\mathrm{HYB}^b_0(n,z)$ and $\mathrm{HYB}^b_1(n,z)$ are indistinguishable.

Claim 3.7. *For each $b \in \{0,1\}$, the following hold.*

- *The probability that $\mathcal{A}$ cheats in $H^b_1(n,z)$ is negligible.*
- *$\{HYB^b_0(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ and $\{HYB^b_1(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ are computationally indistinguishable.*

Proof. First, we show that $\mathcal{A}$ cheats in $H^b_1(n,z)$ with negligible probability for each $b \in \{0,1\}$. Roughly speaking, this follows from the non-malleability of NMCom: Since $H^b_1(n,z)$ differs from $H^b_0(n,z)$ only in the value committed to in NMCom in the left session, the value that $\mathcal{A}$ commits to by using NMCom in the right session of $H^b_1(n,z)$ is indistinguishable from the value that $\mathcal{A}$ commits to by using NMCom in the right session of $H^b_0(n,z)$; hence, from Claim 3.6, the probability that $\mathcal{A}$ cheats in $H^b_1(n,z)$ is negligible. We remark that since the left session in $H^b_1(n,z)$ involves the brute-force extraction of CECom in Stage 1, in the formal argument given below we consider a hybrid experiment in which brute-force extraction is replaced with the rewinding extraction. Since we want to use the non-malleability of NMCom, this extraction is performed in such a way that NMCom in the right session is not rewound. The formal argument is given below.

Assume for contradiction that there exists $b \in \{0,1\}$ such that $\mathcal{A}$ cheats in $H^b_1(n,z)$ with non-negligible probability. Fix any such b. To derive a contradiction, we consider the following hybrid experiment for $i \in \{0,1\}$.

Hybrid $G^b_i(n,z)$ is the same as $H^b_i(n,z)$ except for the following.

- In Stage 1 of the left session, the committed value r of the CECom commitment is extracted by using the extractability of CECom instead of by brute force. Furthermore, this extraction is performed in such a way that the NMCom commitment in the right session is not rewound (see Section 3.5.2).
- $G^b_i(n,z)$ terminates immediately after NMCom ends in Stage 3 of the right session.

From the soundness of ZKArg, the CECom commitment in Stage 1 of the left session is valid when the ZKArg proof in Stage 1 of the left session is accepted. Hence, when Stage 3 is executed in the left session, the value extracted from the CECom commitment in Stage 1 is equal to its (unique) committed value. Since the only difference from $G^b_i(n,z)$ and $H^b_i(n,z)$ is how r is extracted, the view of $\mathcal{A}$ in $G^b_i(n,z)$ is statistically close to that in $H^b_i(n,z)$. Therefore, $\mathcal{A}$ cheats in $G^b_0(n,z)$ with negligible probability from Claim 3.6, and $\mathcal{A}$ cheats in $G^b_1(n,z)$ with non-negligible probability from our hypothesis.

We then derive a contradiction by constructing an adversary $\mathcal{M}$ that breaks the non-malleability of NMCom. Externally, $\mathcal{M}$ interacts with a committer and a receiver of NMCom: It sends 0^n and $r \in \{0,1\}^n$ to the committer and receives a NMCom commitment in which either 0^n or r is committed to; at the same time, it sends a NMCom commitment to the receiver. Internally, $\mathcal{M}$ invokes $\mathcal{A}$ and emulates $G_0^b(n,z)$ for $\mathcal{A}$ honestly except for the following.

- After r is extracted in Stage 1 of the left session, $\mathcal{M}$ sends 0^n and r to the external committer.
- In Stage 3 of the left session, $\mathcal{M}$ forwards the NMCom commitment from the external committer to internal $\mathcal{A}$.
- In Stage 3 of the right session, $\mathcal{M}$ forwards the NMCom commitment from the internal $\mathcal{A}$ to the external receiver.

From the construction, $\mathcal{M}$ perfectly emulates $G_0^b(n,z)$ when it receives a NMCom commitment to 0^n, and it perfectly emulates $G_1^b(n,z)$ when it receives a NMCom commitment to r. Hence, when $\mathcal{M}$ receives a NMCom commitment to 0^n, internal $\mathcal{A}$ cheats with negligible probability, and when $\mathcal{M}$ receives a NMCom commitment to r, internal $\mathcal{A}$ cheats with non-negligible probability. Then, since the cheating of $\mathcal{A}$ is efficiently recognizable given the view of $\mathcal{M}$ and the committed value of the NMCom commitment in the right session, $\mathcal{M}$ breaks the non-malleability of NMCom.

Next, we show that $\{\mathrm{HYB}_0^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ and $\{\mathrm{HYB}_1^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ are computationally indistinguishable. Roughly speaking, this indistinguishability follows from the hiding of NMCom: Since $\mathcal{A}$ cheats only with negligible probability both in $H_0^b(n,z)$ and in $H_1^b(n,z)$, the CECom commitment in Stage 2 is valid in the accepted right session in both hybrids; hence the committed-value oracle can be efficiently emulated by extracting the committed value of the CECom commitment in Stage 2, and thus the indistinguishability follows from the hiding property of NMCom. Here, since we want to use the hiding property of NMCom, the extraction from CECom is performed in such a way that NMCom in the left session is not rewound. The formal argument is given below.

Assume for contradiction that there exists b such that $\{\mathrm{HYB}_0^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ and $\{\mathrm{HYB}_1^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ are distinguishable. Fix any such b. From Claim 3.6 and what is shown above, $\mathcal{A}$ cheats only with negligible probability both in $H_0^b(n,z)$ and in $H_1^b(n,z)$. Hence, from the soundness of WIProof, the CECom commitment in Stage 2 is invalid in the accepted right session only with negligible probability. Therefore, there exists a polynomial $p(\cdot)$ such that for infinitely many n, there exists $z \in \{0,1\}^*$ such that (i) $\mathrm{HYB}_0^b(n,z)$ and $\mathrm{HYB}_1^b(n,z)$ are distinguishable with advantage $1/p(n)$ and (ii) the CECom commitment in Stage 2 of the right session is invalid in the accepted right session with probability at most $1/3p(n)$ in both $H_0^b(n,z)$ and $H_1^b(n,z)$. Fix any such n and z. From an average argument, there exists a partial transcript ρ of $H_0^b(n,z)$ up until the end of Stage 1 of the left session such that under the condition that a prefix of the transcript is ρ, both of the above (i) and (ii) hold. Let r be the value that is committed to in Stage 1 of the left session in ρ. (If the committed value is not uniquely determined, r is a random value.) We consider the following two cases.

Case 1. Stage 2 of the right session has already started in ρ. Since the committed value of a CECom commitment is determined by the first message, ρ uniquely determined the committed value $\widetilde{v}$ of the CECom commitment in Stage 2 of the right session. Notice that given ρ, r, and $\widetilde{v}$ as auxiliary input, $H_0^b(n,z)$ and $H_1^b(n,z)$ can be executed from ρ in polynomial time. Hence, we can derive a contradiction by considering an adversary that breaks the hiding property of NMCom by internally emulating $H_0^b(n,z)$ from ρ and forwarding a NMCom commitment from the external committer (who commits to either 0^n or r) to internally emulated $\mathcal{A}$.

Case 2. Stage 2 of the right session starts after ρ. We consider the following hybrid experiment.

In Hybrid $F_i^b(n,z)$, $H_i^b(n,z)$ is executed from ρ honestly except for the following.

- In the left session, brute-force extraction of r is not performed, and hardwired r is used.
- In Stage 2 of the right session, the committed value $\widetilde{v}$ of the CECom commitment is extracted by using the extractability of CECom in such a way that NMCom in Stage 3 is not rewound in the left session.
- At the end of the right session, the extracted value $\widetilde{v}$ is returned to $\mathcal{A}$ as the committed value of the right session.

From the definition of ρ, the CECom commitment in Stage 2 of the right session is invalid in the accepted right session with probability at most $1/3p(n)$. Since the output of $F_i^b(n,z)$ differs from that of $H_i^b(n,z)$ only when the correct committed value is not extracted in the accepted right session (which occurs with probability at most $1/3p(n)$ from the above), from our hypothesis, the outputs of $F_0^b(n,z)$ and $F_1^b(n,z)$ are distinguishable with advantage $1/3p(n)$. Then, since $F_0^b(n,z)$ and $F_1^b(n,z)$ differ only in the value committed to in NMCom and since both experiments run in polynomial time, we can derive a contradiction by considering an adversary that internally emulates $F_0^b(n,z)$ and forwards a NMCom commitment from the external committer to internally emulated $\mathcal{A}$. □

In the same way above, we can prove that the outputs of the other neighboring hybrids are also indistinguishable.

Claim 3.8. *For each $b \in \{0,1\}$, the following hold.*

- *The probability that $\mathcal{A}$ cheats in $H_2^b(n,z)$ is negligible.*
- *$\{HYB_1^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ and $\{HYB_2^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ are computationally indistinguishable.*

Proof. This claim can be proven in essentially the same way as Claim 3.7. First, we can show that $\mathcal{A}$ cheats in $H_2^b(n,z)$ only with negligible probability by using the same argument except that we use the non-malleability w.r.t. 4-round protocols of NMCom instead of the non-malleability w.r.t. itself. (Recall that $H_2^b(n,z)$ differs from $H_1^b(n,z)$

only in the witness used in WIProof, which has four rounds.) Next, we can show the indistinguishability between $\{\mathrm{HYB}_1^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ and $\{\mathrm{HYB}_2^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ by using the same argument except that we use the witness indistinguishability of WIProof instead of the hiding property of NMCom. We omit the formal proof. □

Claim 3.9. *For each $b \in \{0,1\}$, the following hold.*

- *The probability that $\mathcal{A}$ cheats in $H_3^b(n,z)$ is negligible.*
- *$\{HYB_2^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ and $\{HYB_3^b(n,z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ are computationally indistinguishable.*

Proof. Like Claim 3.8, this claim can be proven in essentially the same way as Claim 3.7. We remark however that since the round complexity of CECom is much more than four, we need to consider a sequence of intermediate hybrid experiments in which the committed value of ExtCom in CECom are switched one by one. We omit the formal proof. □

This concludes the proof of one-one CCA security.

3.5.1.2 Proof of κ-robustness

We show that there exists a simulator $\mathcal{S}$ such that for any adversary $\mathcal{A}$ that interacts with the committed-value oracle only in a single session, and for any κ-round PPT ITM B, the following are computationally indistinguishable:

- $\left\{\mathsf{output}_{B,\mathcal{A}^O}\left[B(1^n,x,y) \leftrightarrow \mathcal{A}^O(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$
- $\left\{\mathsf{output}_{B,\mathcal{S}^{\mathcal{A}}}\left[B(1^n,x,y) \leftrightarrow \mathcal{S}^{\mathcal{A}}(1^n,x,z)\right]\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^n}$

This can be shown easily by using the argument we used in the proof of one-one CCA security. Roughly, we consider a simulator that emulates O for $\mathcal{A}$ efficiently by extracting the committed value of the CECom commitment in Stage 2 using the robust extractability of CECom in such a way that the interaction with B is not rewound. (Since we set $\ell = \max(\kappa, r_{\mathrm{NM}}, 4) + 1$, such extraction is possible.) To show that this simulator indeed emulates the oracle for $\mathcal{A}$, we need to show that the CECom commitment in Stage 2 is invalid in the accepted right session only with negligible probability. This can be shown by using the argument in the proof of Claim 3.6. Hence, by using this simulator, we can prove the κ-robustness. The formal proof is omitted.

3.5.2 On the Robust Extractability of CECom

In this section, we observe that for any constant $\kappa \in \mathbb{N}$, CECom with parameter $\ell = \kappa+1$ satisfies the following robust extractability property: For any adversarial committer C^* that commits to a value in a single session of CECom and simultaneously participates an arbitrary κ-round protocol Π, the extractor can extract the committed value from C^* without rewinding Π. This property is used to obtain constant-round one-one CCA-secure commitment scheme in Section 3.5.1.

Recall that in CECom, the extractable commitment scheme ExtCom of [PW09] is executed ℓ times in the following schedule (cf. Figure 2.2 in Section 2.3.3).

1. First, the `commit`-stage messages of all the sessions (of ExtCom) are exchanged in parallel.

2. Subsequently, the `challenge`-stage message and the `reply`-stage message of the i-th session are exchanged for each $i \in [\ell]$ in sequence.

Let us call the pair of the `challenge`-stage message and the `reply`-stage message of a ExtCom commitment a *slot*. Since the committed value of a ExtCom commitment can be extracted by rewinding the slot and obtaining a new pair of the `challenge`-stage message and the `reply`-stage message (cf. Figure 2.1 in Section 2.3.2), the committed value of a CECom commitment can be extracted by rewinding any of the ℓ slots.

Consider the following extractor E against any adversarial committer C^*. Externally, E participates in a κ-round protocol Π. Internally, E invokes C^* and forwards all messages of Π from the external party to internal C^* and vice verse; additionally, E interacts with C^* in a session of CECom as an honest receiver. (Without loss of generality, we assume that after C^* sends a message of Π [resp., a message of CECom], C^* immediately receives the next message of Π [resp., the next message of CECom].) When the session of CECom ends, E extracts the committed value of the session by rewinding C^* in a slot that does not "interleave" with any message of Π (i.e, a slot such that C^* does not exchange any message of Π after receiving the `challenge` message of the slot until it sends the `reply`-message of the slot; notice that such a slot always exists because there are $\ell = \kappa + 1$ sequential slots). Specifically, E continues to rewind such a slot until it obtains a new pair of the `challenge`-stage message and the `reply`-stage message. If C^* requires a message of Π after being rewound, E cuts off the execution of C^* immediately and rewinds C^* again. After obtaining a new pair of the `challenge`-stage message and the `reply`-stage message, it extracts the committed value by using them.

From the construction, E perfectly emulates the view of C^* and does not rewind the external protocol Π. Also, from the extractability of ExtCom, the extraction fails only with negligible probability. Hence, it remains to show that E runs in (expected) polynomial time. This can be shown easily by using the standard "$p \times 1/p$" argument as follows. For any $i \in [\ell]$ and any partial view ρ_i of C^* from which the i-th slot starts, let prefix_{ρ_i} be the event that in the execution of E, the view of internal C^* up until the beginning of the i-th slot is ρ_i. Let T_i be the random variable representing the number of rewinding in the i-th slot in E, and let p_{ρ_i} be the probability that under the condition that prefix_{ρ_i} occurs, the i-th slot is accepting and it does not interleave with any message of Π. We then have

$$\mathrm{E}\left[T_i \mid \mathsf{prefix}_{\rho_i}\right] \leq p_{\rho_i} \cdot 1/p_{\rho_i} = 1$$

for any ρ_i. Thus, we have

$$\mathrm{E}\left[T_i\right] = \sum_{\rho_i} \mathrm{E}\left[T_i \mid \mathsf{prefix}_{\rho_i}\right] \Pr\left[\mathsf{prefix}_{\rho_i}\right] \leq \sum_{\rho_i} \Pr\left[\mathsf{prefix}_{\rho_i}\right] \leq 1 \ .$$

Hence, from the linearity of expectation, the expected number of rewinding of C^* in the execution of E is at most ℓ, and thus the expected running time of E can be bounded by a polynomial.

Chapter 4

Constant-round Leakage-resilient Zero-knowledge from Collision Resistance

In this chapter, we show our second result: A constant-round leakage-resilient zero-knowledge argument based on collision-resistant hash functions.

4.1 Background

As one can see in Definition 2.11, the zero-knowledgeness of interactive proofs/arguments is defined in the setting where an adversarial verifier obtain information about honest parties' internal states only though the proofs that they receive from the provers.

Recently, Garg et al. [GJS11] introduced a new notion of zero-knowledgeness called *leakage-resilient zero-knowledge* (LRZK), which is, roughly speaking, a notion of zero-knowledgeness in the setting where adversarial verifiers can obtain arbitrary leakage on the entire state of the honest prover (including the witness and the randomness) during the entire protocol execution. LRZK is motivated by the studies of *side-channel attacks* (e.g., [Koc96, AK96, QS01]), which demonstrated that adversaries might be able to obtain leakage of honest parties' secret states by attacking physical implementations of cryptographic algorithms.

Informally speaking, LRZK requires that the protocol does not reveal anything beyond the validity of the statement *and the leakage that the adversary obtained*. More formally, LRZK is defined as follows. In the definition of LRZK, the cheating verifier is allowed to make arbitrary number of *leakage queries* during the interaction with an honest prover, where each leakage query f is answered by $f(w, \mathsf{tape})$ for the witness w and the randomness tape that the honest prover generated thus far. On the other hand, the simulator is allowed to make queries to the *leakage oracle* $\mathcal{L}_w$, which is parametrized by the witness w of the honest prover and outputs $f(w)$ on input any function f. LRZK is then defined by requiring that for any cheating verifier V^* there exists a simulator $\mathcal{S}$ such that for any $\ell \in \mathbb{N}$, when V^* obtains ℓ bits of leakage of the prover's state via leakage queries, $\mathcal{S}$ can simulate the view of V^* by obtaining ℓ bits of leakage of the

witness via queries to the leakage oracle $\mathcal{L}_w$.[21]

In [GJS11], Garg et al. showed a proof system that satisfies a weaker notion of LRZK called $(1+\epsilon)$-LRZK. Specifically, they showed that for any $\epsilon > 0$, there exists a proof system such that when V^* obtains ℓ bits of leakage from the prover, a simulator can simulate the verifier's view by obtaining at most $(1+\epsilon)\cdot\ell$ bits of leakage from $\mathcal{L}_w$. The round complexity of this protocol is at least $\omega(\log n)/\epsilon$, and its security is proven under a standard general assumption (the existence of statistically hiding commitment schemes that are public-coin w.r.t. the receivers). Garg et al. also showed that their protocol can be used to relax the assumption on the "tamper-proofness" of hardware tokens that are used in the design of various cryptographic protocols.

A natural question left open by [GJS11] is whether we can construct a LRZK protocol without weakening the security requirement. That is, the question is whether we can reduce ϵ to 0 in the protocol of [GJS11]. This question is important because, although $(1+\epsilon)$-LRZK is useful in several applications, $(1+\epsilon)$-LRZK does not guarantee sufficient level of security in many applications. (In fact, since $(1+\epsilon)$-LRZK allows the simulator to obtain strictly more leakage than the adversary, $(1+\epsilon)$-LRZK protocols can potentially reveal secret information in addition to the leakage.) The question of reducing ϵ to 0 is also of theoretical interest because reducing ϵ to 0 is optimal in the sense that λ-LRZK for $\lambda < 0$ is impossible to achieve in the plain model [GJS11].

Recently, this open question was solved affirmatively by Pandey [Pan14], who constructed the first LRZK argument system by using the DDH assumption and collision-resistant hash functions. Pandey's protocol has a desirable property that it has only constant number of rounds; hence, his result implies that asymptotically optimal round complexity is achievable even in the presence of leakage.

A question that is explicitly left open by Pandey [Pan14, Section 1] is whether we can construct LRZK protocols under a standard *general* assumption. In fact, although the protocol of [Pan14] is superior to the protocol of [GJS11] in terms of both leakage resilience (LRZK v.s. $(1+\epsilon)$-LRZK) and round complexity (constant v.s. $\omega(\log n)/\epsilon$), the assumption of the former is seemingly much stronger than that of the latter (the DDH assumption v.s. the existence of statistically hiding commitment schemes that are public-coin w.r.t. the receivers, which is implied by, say, the existence of collision-resistant hash function family or even the existence of one-way functions[22]).

> **Question.** *Can we construct a (constant-round) leakage-resilient zero-knowledge protocol under standard general assumptions?*

4.1.1 Our Results

In this chapter, we answer the above question affirmatively by constructing a LRZK protocol from collision-resistant hash functions (CRHFs). Like the protocol of [Pan14],

[21] In [OPV15], it is pointed out that nowadays *leakage tolerance* is the commonly accepted term for this security notion. Nevertheless, in this thesis we use the term "leakage resilience" for this security notion for consistency with previous works [GJS11, Pan14].

[22] A constant-round one can be constructed from collision-resistant hash functions [NY89, DPP98] and a polynomial-round one can be constructed from one-way functions [HNO+09].

our protocol has only constant number of rounds. Also, our protocol has an additional property that it is public coin (w.r.t. the verifier).

Main Theorem. *Assume the existence of collision-resistant hash function family. Then, there exists a constant-round public-coin leakage-resilient zero-knowledge argument for $\mathcal{NP}$.*

We notice that the existence of LRZK protocols under CRHFs is somewhat surprising because the only known LRZK protocol [Pan14] crucially relies on the secure two-party computation protocol of Yao [Yao86], which requires an assumption that is seemingly stronger than the existence of CRHFs (namely the existence of oblivious transfer protocols). One of our technical novelties is the construction of LRZK without Yao's protocol.

Simultaneously leakage-resilient zero-knowledge. Our protocol has an additional property that it is *simultaneously leakage-resilient zero-knowledge* [GJS11], meaning that not only zero-knowledgeness but also soundness holds in the presence of leakage. The *leakage-resilient (*LR*) soundness* (i.e., soundness in the presence of leakage) of our protocol follows immediately from its public-coin property. In fact, any public-coin interactive proof/argument system is LR sound for arbitrary amount of leakage of the verifier because the verifier has no secret state in public-coin protocols.

To the best of our knowledge, our protocol is the first simultaneously LRZK protocol. The $(1+\epsilon)$-LRZK protocol of Garg et al. [GJS11] is LR sound in a weak sense—it is LR sound when there is an a-priori upper bound on the amount of leakage—but is not LR sound when the amount of leakage is unbounded,[23] and similarly, the LRZK protocol of Pandey [Pan14] is also not LR sound with unbounded amount of leakage. In contrast, our protocol is sound even when cheating verifiers obtain arbitrary amount of leakage on the secret state of the verifier.

A summary of the previous results and ours is given in Table 4.1. In the table, "bounded-LR sound" means that the soundness holds when there is an a-priori upper bound on the amount of leakage from the verifier.

4.1.2 Open Questions

Reducing assumption to one-way functions. An important open question is whether we can construct constant-round LRZK argument systems under the existence of one-way functions.

We notice that solving this question affirmatively seems to require an advancement on "straight-line" simulation techniques (i.e., techniques that do not use rewinding). This is because, as will become clear in Section 4.2, constant-round LRZK seems to require straight-line simulation, and currently the only known

[23] This is because in the protocol of [GJS11], the verifier commits to the challenge bits of Blum's Hamiltonicity protocol in advance and hence an cheating prover can easily break the soundness by obtaining the challenge bits via leakage.

	ZKness	Soundness	#(round)	Assumptions
[GJS11]	$(1+\epsilon)$-LR	bounded-LR	$\mathsf{poly}(n) + \omega(\log n)/\epsilon$	OWFs
			$\omega(\log n)/\epsilon$	CRHFs
[Pan14]	LR	-	$O(1)$	DDH + CRHFs
This work	LR	LR	$O(1)$	CRHFs

Table 4.1: Summary of the results on LRZK protocols. In the table, "LR" stands for "leakage-resilient." The round complexity of the protocol of [GJS11] depends on the assumption that is used to instantiate the underlying statistically-hiding commitment scheme; in particular, when only one-way functions (OWFs) are used, there is a polynomial additive overhead because statistically hiding commitment schemes currently require polynomial number of rounds in this case [HNO+09].

straight-line simulation technique, the one by Barak [Bar01], requires collision-resistant hash functions.[24]

Constructing LRZK proof system. Another open question is whether we can construct LRZK proof systems (instead of argument systems).

We notice that solving this question affirmatively also seems to require an advancement on straight-line simulation techniques. This is because the straight-line simulation technique by Barak [Bar01] is currently inherently only computationally sound.

4.1.3 Related Works

The works relevant to ours are the works that study interactive protocols in the presence of arbitrary leakage in the models other than the plain model. These works include the works about leakage-tolerant UC-secure protocols in the CRS model [BCH12], non-transferable interactive proof systems in the CRS model with leak-free input encoding/updating phase [AGP14], and secure computation protocols in the CRS model with leak-free preprocessing/input-encoding phase and constant fraction of honest parties [BGJK12, BGJ+13, BDL14]. We remind the readers that, like [GJS11, Pan14], this work considers LRZK protocols in the plain model without any leak-free phase.

In [OPV15], Ostrovsky et al. showed an impossibility result about black-box LRZK (and leakage-resilient MPC for several functionalities) in the model with only leak-free input-encoding phase (i.e., without CRS and preprocessing). We notice that this impossibility result does not contradict our result since the definition of LRZK in [OPV15] is different from the one we use (i.e., the definition given by [GJS11]). Specifically, in the definition of [OPV15], the simulator is not allowed to obtain any leakage, whereas in the definition that we use, the simulator can obtain the same amount of leakage as the cheating verifier. (In other words, Ostrovsky et al. [OPV15] considers leakage resilience whereas we consider leakage tolerance; see Footnote 21.)

[24] In [CPS13], Chung et al. showed that the simulation technique of Barak can be modified so that it requires only one-way functions. However, the simulation technique of Chung et al. involves rewinding of the adversary and therefore is no longer straight-line simulation.

4.1.4 Outline

In Section 4.2, we give an overview of our techniques. In Section 4.3, we give the notations and definitions that are used specifically in this chapter. In Section 4.4, we show two new building blocks that we use in our LRZK protocol. In Section 4.5, we describe our LRZK protocol and prove its security.

4.2 Overview of Our Techniques

4.2.1 Previous Techniques

Since our techniques rely on the techniques that are used in the previous LRZK protocols of [GJS11, Pan14], we start by recalling these protocols.

Protocol of [GJS11].

In [GJS11], Garg et al. constructed a $(1 + \epsilon)$-leakage-resilient zero-knowledge proof system from a statistically hiding commitment scheme that is public-coin w.r.t. the receiver. That is, they constructed a proof system such that, when V^* obtains ℓ bits of leakage from the prover, its view can be simulated by obtaining at most $(1 + \epsilon) \cdot \ell$ bits of leakage from $\mathcal{L}_w$.

A key idea behind the protocol of [GJS11] is to give the simulator two independent ways of cheating—one for simulating prover's messages and the other for simulating leakages. Concretely, Garg et al. constructed their protocol by combining two well-known techniques of constant-round zero-knowledge protocols—the technique of [GK96a] that requires the verifier to commit to its challenges in advance and the technique of [FS90b] that uses equivocal commitment schemes. They then proved the security by considering a simulator that simulates the prover's messages by extracting the challenges and simulates the leakages by using the equivocality of the underlying commitment scheme.

In more details, the protocol of [GJS11] consists of the following two phases. In the first phase, the verifier uses an extractable commitment scheme to commit to a challenge string ch of Blum's Hamiltonicity protocol as well as trapdoor information td of an equivocal commitment scheme.[25] In the second phase, the prover and the verifier execute Blum's Hamiltonicity protocol that is instantiated with the equivocal commitment scheme. In simulation, the simulator extracts ch and td in the first phase and then simulates the prover's messages and the leakages in the second phase by using the knowledge of ch and td in the following way. (For simplicity, we assume that Blum's protocol is executed only once instead of many times in parallel.)

- When the extracted challenge ch is 0, the simulator commits to a randomly permuted graph of statement G, and after V^* decommits the challenge ch (which must be 0), the simulator decommits the commitment to the permuted graph of G.

[25] Actually, there is a coin-tossing protocol that determines the parameter of the equivocal commitment, and td is the trapdoor for biasing the outcome of the coin-tossing.

Notice that the simulator does exactly the same things as an honest prover. Hence, the simulator can simulate prover's randomness tape easily and therefore can answer any leakage query f from V^* by querying $f(\cdot, \mathsf{tape})$ to $\mathcal{L}_w$.

- When the extracted challenge ch is 1, the simulator commits to a randomly chosen cycle graph H at the beginning and then partially decommits it in the last step so that only the edges on the cycle are revealed.

  When V^* makes a leakage query, the simulator answers it by using the fact that, given w and td, it is possible to compute randomness that "explains" the commitment to H as a commitment to a permuted graph of G. Specifically, the simulator answers a leakage query f from V^* by querying $\mathcal{L}_w$ the following function $\widetilde{f}(\cdot)$.

  1. On input w, function $\widetilde{f}$ first computes a permutation π that maps the Hamiltonian cycle w in G to the cycle in H (i.e., computes π such that $\pi(G)$ has the same cycle as H).
  2. Then, by using equivocality[26] with trapdoor td, it computes randomness tape that explains the commitment to H as a commitment to $\pi(G)$ (i.e., it computes tape such that committing to $\pi(G)$ with randomness tape will generate the same commitment as the one that the simulator has sent to V^* by committing to H).
  3. Finally, it outputs $f(w, \mathsf{tape})$.

  Notice that since $\pi(G)$ has the same cycle as H, the simulated leakages (from which V^* may be able to compute $\pi(G)$) are consistent with the decommitted cycle of H in the last step.

We remark that the reason why the protocol of [GJS11] satisfies only $(1+\epsilon)$-LRZK (rather than standard LRZK) is that the extraction of ch and td involves the rewinding of V^*. Indeed, if V^* makes new leakage queries after being rewound, the simulator need to obtain new leakages from $\mathcal{L}_w$, so the simulator need to obtain more bits of leakage than V^*. From this observation, it seems that to achieve LRZK, we need to avoid the use of rewinding simulation techniques.

Protocol of [Pan14].

In [Pan14], Pandey constructed a constant-round LRZK argument system under the DDH assumption. Roughly speaking, Pandey's idea is to replace the rewinding simulation technique in the protocol of [GJS11] with the "straight-line" simulation technique of Barak [Bar01]. In particular, Pandey replaced the first phase of the protocol of [GJS11] with the following one.

1. First, the prover and the verifier execute an encrypted version of so called *Barak's preamble* [Bar01, PR05b, PR05a], which determines a "fake statement" that is false except with negligible probability.

[26] What is actually used here is *adaptive security*, which guarantees that for each underlying commitment, it is possible to compute randomness tape_0 and tape_1 such that tape_b explains the commitment as a commitment to b for each $b \in \{0, 1\}$.

2. Next, the prover and the verifier execute Yao's garbled circuit protocol [Yao86] in which the prover can obtain ch and td only when it has a valid witness for the fake statement.

From the security of the encrypted Barak's preamble, no cheating prover can make the fake statement true; hence, ch and td are hidden from the cheating prover. In contrast, a non-black-box simulator can make the fake statement true by using the knowledge of the code of the verifier; hence, the simulator can obtain ch and td without rewinding V^*. An issue is that, to guarantee leakage resilience, it is required that Yao's protocol is executed in a way that all messages from the prover are pseudorandom (since otherwise it is hard to simulate randomness that explains the simulated prover's messages as honest prover's messages during the simulation of the leakages). Since Yao's protocol involves executions of an oblivious transfer protocol (in which the prover behaves as a receiver), this property is not easy to satisfy. Pandey solved this problem by using the DDH assumption, under which there exists an oblivious transfer protocol such that all messages from the receiver are indistinguishable from random group elements.

4.2.2 Our Techniques

The reason why the protocols of [GJS11, Pan14] either guarantee only weaker security or rely on a stronger assumption is that the simulation involves extraction from V^*. Indeed, in [GJS11] the simulator need to obtain more amount of leakage than V^* because it rewinds V^* during extraction, and in [Pan14] the DDH assumption is required because Yao's protocol is used for extraction.

Based on this observation, our strategy is to modify the protocols of [GJS11, Pan14] so that no extraction is required in simulation. We first remove the extraction of trapdoor td and next remove the extraction of challenge ch. We remark that the latter is much harder than the former.

Removing Extraction of Trapdoor td.

We first modify the protocols of [GJS11, Pan14] so that leakages can be simulated without extracting the trapdoor td of an equivocal commitment scheme.

Our main tool is Hamiltonicity commitment scheme $\mathsf{H\text{-}Com}$ [FS90b, CLOS02], which is a well-known instance-dependent equivocal commitment scheme based on Blum's Hamiltonicity protocol. $\mathsf{H\text{-}Com}$ is parametrized by a graph G with $q = \mathsf{poly}(n)$ vertices. To commit to 0, the committer chooses a random permutation π and commits to the adjacency matrix of $\pi(G)$ using any commitment scheme Com; in the decommit phase, the committer reveals π and decommits all the entries of the matrix. To commit to 1, the committer commits to the adjacency matrix of a random q-cycle graph; in the decommit phase, the committer decommits only the entries that corresponds to the edges on the cycle. $\mathsf{H\text{-}Com}$ satisfies equivocality when G has a Hamiltonian cycle; this is because after committing to 0, the committer can decommit it to both 0 and 1 given a Hamiltonian cycle w in G.

Given $\mathsf{H\text{-}Com}$, we remove the extraction of td by combining $\mathsf{H\text{-}Com}$ with an encrypted variant of Barak's preamble. Specifically, we replace the equivocal commit-

ment scheme in the protocols of [GJS11, Pan14] with H-Com that depends on the fake statement G' that is obtained by the encrypted Barak's preamble. From the security of Barak's preamble, any cheating prover cannot make G' true and hence cannot use the equivocality of H-Com, whereas the simulator can make G' true and hence can use the equivocality of H-Com as desired.

Remark 4.1. As observed in [Pan14], it is not straightforward to use the encrypted Barak's preamble in the presence of leakage. Roughly speaking, in the encrypted Barak's preamble, the prover commits to its messages instead of sending them in clear, and in the proof of soundness, it is required that the prover's messages are extractable from the commitments. The problem is that it is not easy to guarantee this extractability in the presence of leakage (this is because the prover's messages are typically not pseudorandom in the techniques of extractability). Pandey [Pan14] solved this problem by having the prover use a specific extractable commitment scheme based on the DDH assumption. In this thesis, we solve this problem by having the prover use a commitment scheme that satisfies only very weak extractability but the prover's messages of which are pseudorandom and the security of which is based on the existence of one-way functions (which is implied by the existence of CRHFs).[27] For details, see Section 4.4.1.

Removing Extraction of Challenge *ch*.

Next, we modify the protocols of [GJS11, Pan14] so that prover's messages can be simulated without extracting the challenge *ch* of Hamiltonicity protocol. Surprisingly, we can do this without using any heavy machinery; all that is required is a clever use of the Hamiltonicity protocol.

We first notice that, although the simulator can use equivocality without extraction as shown above, it is not easy for the simulator to use equivocality for simulating prover's messages. This is because if the leakages to V^* includes the randomness that is used for some commitments, V^* may be able to determine their committed values from the leakages and therefore may be able to detect equivocation on them.

As our main technical tool, then, we introduce a specific instance-dependent equivocal commitment scheme GJS-Com that we obtain by viewing the technique of [GJS11] on Hamiltonicity protocol in the context of H-Com. Recall that in [GJS11], Garg et al. use Blum's Hamiltonicity protocol that is instantiated with an equivocal commitment scheme. Here, we use Hamiltonicity commitment scheme H-Com that is instantiated with an equivocal commitment scheme (i.e., we use H-Com in which the adjacency matrix is committed to by an equivocal commitment scheme). The equivocal commitment scheme that we use here is, as above, H-Com that depends on the fake statement generated by the encrypted Barak's preamble.[28] Hence, the commitment scheme GJS-Com is a version of H-Com that is instantiated by using H-Com itself as the underling commitment scheme.[29] GJS-Com depends on two statements of the Hamiltonicity problem: The "outer" H-Com (the H-Com that is implemented with H-Com) depends on

[27] This extractability is used only in the proof of soundness. Hence, the proof of zero-knowledgeness works even in the presence of this extractable commitment scheme.

[28] Actually, we use an adaptively secure H-Com [CLOS02, LZ11]. See Footnote 26.

[29] In the "inner" H-Com, the underlying commitment scheme is Com as before.

the real statement G, and the "inner" H-Com (the H-Com that is used to implement H-Com) depends on the fake statement G'. GJS-Com inherits equivocality from the outer H-Com, i.e., given a witness for the real statement G, a GJS-Com commitment to 0 can be decommitted to both 0 and 1.

Since GJS-Com is obtained by viewing the technique of [GJS11] in the context of H-Com, we can see that GJS-Com satisfies a property that is useful for proving LRZK property. We first observe that given GJS-Com, the second phase of the LRZK protocol of [GJS11] (i.e., Blum's Hamiltonicity protocol phase) can be viewed as follows.

1. The prover commits to 0 by using GJS-Com.
2. The verifier reveals the challenge $ch \in \{0, 1\}$ that is committed to in the first phase.
3. When $ch = 0$, the prover decommits the GJS-Com commitment to 0 honestly, and when $ch = 1$, the prover decommits it to 1 by using the equivocality with the knowledge of Hamiltonian cycle w in G.

Also, the simulation of the prover's messages in [GJS11] can be viewed as follows: first, ch is extracted in the first phase; then, when $ch = 0$, the simulator performs honestly in the second phase (i.e., commits to 0 by using GJS-Com and then decommits to 0) while when $ch = 1$, the simulator commits to 1 by using GJS-Com and then decommits to 1. Now, when the second phase of the protocol of [GJS11] and the simulation of the prover's messages are viewed in this way, the key property that is used in the simulation of the leakages in [GJS11] is the following.

- Given a Hamiltonian cycle in G and that in G', a GJS-Com commitment to 1 (in which a random cycle graph is committed) can be "explained" as a commitment to 0 (in which a permutation of G is committed) by using the equivocality of the inner H-Com. Furthermore, even after being explained as a commitment to 0, the commitment can later be decommitted to 1 in a consistent way with the explained randomness (cf. function $\widetilde{f}$ in Section 4.2.1).

Because of this property, even when the simulator commits to 1 instead of 0 using GJS-Com to simulate the messages, the simulator can still answer any leakage query from V^* consistently with the simulated messages—to answer to a leakage query f from V^*, the simulator queries the following function $\widetilde{f}$ to the leakage oracle $\mathcal{L}_w$: On input w, $\widetilde{f}$ computes randomness tape that explains the commitment to 1 as a commitment to 0, and then it outputs $f(w, \mathsf{tape})$.

A problem of this property is that it can be used only in a very limited situation. Specifically, this property can be used only when the simulator knows which GJS-Com commitment will be decommitted to 1 (this is because this property can be used only when the simulator gives GJS-Com commitments to 1, and the simulator cannot decommit them to 0 because it does not know the witness for the real statement G), and this is the reason why the extraction of ch is required in the simulation strategy of [GJS11, Pan14]. Hence, to remove the extraction of ch, we need to use GJS-Com in a way that, given a witness for the fake statement, the simulator can predict which value each GJS-Com commitment will be decommitted to.

Then, our key observation is that we can use this property if we use GJS-Com to implement the Hamiltonicity protocol *in which the fake statement is proven*.[30] Concretely, we consider the following protocol.

1. The prover and the verifier execute an encrypted variant of Barak's preamble. Let G' be the fake statement and q' be the number of the nodes in G'.

2. (a) The prover commits to a $q' \times q'$ zero matrix by using GJS-Com.

   (b) The verifier sends a challenge $ch \in \{0, 1\}$.

   (c) When $ch = 0$, the prover sends a random permutation π over G' to the verifier and then decommit the GJS-Com commitments to the adjacency matrix of $\pi(G')$ by using the equivocality of GJS-Com with the knowledge of a witness for the real statement.

   When $ch = 1$, the prover chooses a random q'-cycle graph H and decommits some of the GJS-Com commitments to 1 by using the equivocality of GJS-Com so that the decommitted entries of the matrix correspond to the cycle in H.

   (d) When $ch = 0$, the verifier verifies whether the decommitted graph is $\pi(G')$. When $ch = 1$, the verifier verifies whether the decommitted entries corresponds to a q'-cycle in a graph.

Since any charting prover cannot make the fake statement G' true, GJS-Com is statistically binding when the real statement G is false, so soundness can be proven as for the original version of Blum's Hamiltonicity protocol. In contrast, the simulator can cheat in Barak's preamble and learn a Hamiltonian cycle w' in the fake statement G', so it can simulate the prover's messages by "honestly" proving the fake statement, i.e., by committing to $\pi(G')$ in step 2(a) for a randomly chosen π and then revealing the entire graph $\pi(G')$ or only the cycle $\pi(w')$ depending on the value of ch. Furthermore, since in step 2(a) the simulator do know which value each GJS-Com commitment will be decommitted to (the commitments to the edges on $\pi(w')$ will be always decommitted to 1 and others will be decommitted honestly or will not be decommitted), the simulator can simulate the leakage in the same way as in the protocol of [GJS11] by using the property of GJS-Com described above—that is, by querying $\mathcal{L}_w$ a function that simulates leakage by "explaining" each commitment to 1 as a commitment to 0.

Since Barak's preamble is based on the existence of CRHFs and has constant rounds, our protocols is based on the existence of CRHFs and has constant rounds. This completes the overview of our techniques.

[30] Hence, we use Hamiltonicity protocol recursively *three times*: We instantiate Hamiltonicity commitment with Hamiltonicity commitment to obtain GJS-Com, and then instantiate Blum's Hamiltonicity protocol with GJS-Com.

4.3 Preliminaries

4.3.1 Notations

We use $\mathbf{L}_{\mathrm{HC}}$ to denote the languages of the Hamiltonian graphs. For any $G \in \mathbf{L}_{\mathrm{HC}}$, we use $\mathbf{R}_{\mathrm{HC}}(G)$ to denote the set of the Hamiltonian cycles in G. Generally, for any language $\mathbf{L}$ and any instance $x \in \mathbf{L}$, we use $\mathbf{R}_{\mathbf{L}}(x)$ to denote the set of the witnesses for $x \in \mathbf{L}$.

We use Com to denote Naor's 2-round statistically binding commitment scheme, and use $\mathsf{Com}_r(\cdot)$ to denote an algorithm that, on input $m \in \{0,1\}^*$, computes a commitment to $m \in \{0,1\}^*$ by using Naor's commitment scheme with the first-round message being r (cf. Section 2.3.1). We remark that $\mathsf{Com}_r(\cdot)$ has pseudorandom range; hence, by using a public-coin algorithm $\mathsf{Com}_{\mathrm{pub}}$ that outputs a random $3n\ell$-bit string on input 1^ℓ, we can obtain a "fake commitment" that is indistinguishable from a real commitment to an ℓ-bit string.

In this chapter, we use $\mathsf{Value}(\cdot)$ to denote a function that, on input a commitment (i.e., a transcript in the commit phase), outputs its committed value if it is uniquely determined and outputs $\perp$ otherwise.

4.3.2 Leakage-resilient Zero-knowledge

In this section, we recall the definition of leakage-resilient zero-knowledgeness [GJS11]. For convenience, we use a slightly different formulation of the definition.

For any interactive proof system $\langle P, V\rangle$, any PPT cheating receiver V^*, any statement $x \in \mathbf{L}$, any witness $w \in \mathbf{R}_{\mathbf{L}}(x)$, and any oracle machine $\mathcal{S}$ called a *simulator*, consider the following two experiments.

$\underline{\mathrm{REAL}_{V^*}(x, w, z)}$

1. Execute $V^*(x, z)$ with an honest prover $P(x, w)$ of $\langle P, V\rangle$.
   During the interaction, V^* can make arbitrary number of adaptive leakage queries on the state of P. A leakage query consists of an efficiently compatible function f_i (described as a circuit) and it is answered with $f_i(w, \mathsf{tape})$, where tape is the randomness used by P so far.
2. Output the view of V^*.

$\underline{\mathrm{IDEAL}_{\mathcal{S}}(x, w, z)}$

1. Execute $\mathcal{S}(x, z)$ with access to a leakage oracle $\mathcal{L}_w$. A query to $\mathcal{L}_w$ consists of an efficiently computable function f and answered with $f(w)$. Let τ be the output of $\mathcal{S}$.
2. If τ is not valid view of V^*, the output of the experiment is $\perp$. Otherwise, let ℓ be the total length of the leakage that V^* obtains in τ. If the total length of the answers that $\mathcal{S}$ obtained from $\mathcal{L}_w$ is larger than ℓ, the output of the experiment is $\perp$. Otherwise, the output is τ.

Let $\mathsf{REAL}_{V^*}(x, w, z)$ be the random variable representing the output of $\mathrm{REAL}_{V^*}(x, w, z)$ and $\mathsf{IDEAL}_{\mathcal{S}}(x, w, z)$ be the random variable representing the output of $\mathrm{IDEAL}_{\mathcal{S}}(x, w, z)$. Then, leakage resilient zero-knowledgeness is defined as follows.

Definition 4.1. *An interactive argument system $\langle P, V\rangle$ for a language $\boldsymbol{L}$ with witness relation $\boldsymbol{R}$ is **leakage-resilient zero knowledge** if for every PPT machine V^* and every sequence $\{w_x\}_{x\in\boldsymbol{L}}$ such that $(x, w_x) \in \boldsymbol{R}_{\boldsymbol{L}}$, there exists a PPT oracle machine $\mathcal{S}$ such that the following hold.*

Indistinguishability condition.

$$\{\mathsf{REAL}_{V^*}(x, w_x, z)\}_{x\in\boldsymbol{L},z\in\{0,1\}^*} \overset{c}{\approx} \{\mathsf{IDEAL}_{\mathcal{S}}(x, w_x, z)\}_{x\in\boldsymbol{L},z\in\{0,1\}^*} \quad .$$

Leakage-length condition. *For every $x \in \boldsymbol{L}$ and $z \in \{0, 1\}^*$,*

$$\Pr\left[\mathsf{IDEAL}_{\mathcal{S}}(x, w_x, z) = \bot\right] = 0 \quad .$$

4.3.3 Hamiltonicity Commitment Scheme

In this section, we recall a well-known instance-dependent commitment scheme H-Com [FS90b, CLOS02] that is based on Blum's zero-knowledge proof for Hamiltonicity.

Commit phase. H-Com is parametrized by a graph G. Let q be the number of its vertices. To commit to 0, the committer chooses a random permutation π over the vertices of G and then commits to the adjacency matrix of $\pi(G)$ by using Naor's commitment scheme Com. To commit to 1, the committer chooses a random q-cycle graph and then commits to its adjacency matrix by using Com.

We use $\mathsf{H\text{-}Com}_{G,r}(\cdot)$ to denote an algorithm that, on input $b \in \{0, 1\}$, computes a commitment to b as above by using r as the first-round message of all the Com commitments.

Decommit phase. When the committer committed to 0, it reveals π, and also reveals all the entries of the adjacency matrix by decommitting all the Com commitments. When the committer committed to 1, it reveals only the entries corresponding to the edges on the q-cycle by decommitting the Com commitments in which these entries are committed.

Security. H-Com is computationally hiding, and it is statistically binding when $G \notin \mathbf{L}_{\mathrm{HC}}$.

Equivocality. When $G \in \mathbf{L}_{\mathrm{HC}}$, a commitment to 0 can be decommitted to 1 given a Hamiltonian cycle $w \in \mathbf{R}_{\mathrm{HC}}(G)$ in G. Specifically, a commitment to 0 can be decommitted to 1 by decommitting the entries that corresponds to the edges on $\pi(w)$ (i.e., the cycle that is obtained by applying π on w).

4.3.4 Adaptive Hamiltonicity Commitment Scheme

In this section, we recall the adaptively secure Hamiltonicity commitment scheme AH-Com, which was used in, e.g., [CLOS02, LZ11].

Commit phase. AH-Com is parametrized by a graph G. Let q be the number of its vertices. To commit to 0, the committer does the same things as in H-Com; i.e., it chooses a random permutation π over the vertices of G and then commits to the adjacency matrix of $\pi(G)$ by using Naor's commitment scheme Com. To commit to 1, the committer chooses a random q-cycle graph and then commits to its adjacency matrix in the following way: For all the entries corresponding to the edges on the q-cycle, it commits to 1 by using Com, and for all the other entries, it simply sends random $3n$-bit strings instead of committing to 0. (Since Com has pseudorandom range, random $3n$-bit strings are indistinguishable from Com commitments; see Section 2.3.1.)

We use $\mathsf{AH\text{-}Com}_{G,r}(\cdot)$ to denote an algorithm that, on input $b \in \{0,1\}$, computes a commitment to b as above by using r as the first-round message of all the Com commitments.

Decommit phase. To decommit, the committer reveals all the randomness used in the commit phase. We use $\mathsf{AH\text{-}Dec}_r(\cdot,\cdot,\cdot)$ to denote an algorithm that, on input c, b, ρ such that $\mathsf{AH\text{-}Com}_r(b;\rho) = c$, outputs a decommitment d as above.

Security. Like H-Com, AH-Com is computationally hiding both when $G \in \mathbf{L}_{\mathrm{HC}}$ and when $G \notin \mathbf{L}_{\mathrm{HC}}$, and it is statistically binding when $G \notin \mathbf{L}_{\mathrm{HC}}$.

Adaptive security. When $G \in \mathbf{L}_{\mathrm{HC}}$, a commitment to 0 can be "explained" as a valid commitment to 1 given a witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$. Specifically, for a commitment c to 0, we can compute ρ such that $\mathsf{AH\text{-}Com}(1;\rho) = c$. This is because commitments to the entries that do not correspond to the edges on $\pi(w)$ are indistinguishable from random strings.

Formally, there exists an algorithm AH-ExplainAsOne such that for security parameter $n \in \mathbb{N}$, graphs $G \in \mathbf{L}_{\mathrm{HC}}$, witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$, and string $r \in \{0,1\}^{3n}$, the following hold.

Correctness. Given witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$ and c, ρ such that $\mathsf{AH\text{-}Com}_{G,r}(0;\rho) = c$, $\mathsf{AH\text{-}ExplainAsOne}_{G,r}$ outputs ρ' such that $\mathsf{AH\text{-}Com}_{G,r}(1;\rho') = c$.

Indistinguishability. Consider the following two probabilistic experiments.

$\underline{\mathrm{Exp}_0^{\mathrm{AH}}(n, G, w, r)}$

```
/* commit to 1 and reveal randomness */
```

1. Computes $c \leftarrow \mathsf{AH\text{-}Com}_{G,r}(1)$.
   Let ρ_1 be the randomness used in AH-Com.
2. Output (c, ρ_1).

$\underline{\mathrm{Exp}_1^{\mathrm{AH}}(n, G, w, r)}$

```
/* commit to 0 and explain it as commitment to 1 */
```

1. Computes $c \leftarrow \mathsf{AH\text{-}Com}_{G,r}(0)$.
   Let ρ_0 be the randomness used in $\mathsf{AH\text{-}Com}$.
   Compute $\rho_1 := \mathsf{AH\text{-}ExplainAsOne}_{G,r}(w, c, \rho_0)$.
2. Output (c, ρ_1).

For each $b \in \{0, 1\}$, let $\mathsf{Exp}_b^{\mathrm{AH}}(n, G, w, r)$ be the random variable representing the output of $\mathrm{Exp}_b^{\mathrm{AH}}(n, G, w, r)$. Then, the following two ensembles are computationally indistinguishable.

- $\left\{\mathsf{Exp}_0^{\mathrm{AH}}(n, G, w, r)\right\}_{n\in\mathbb{N}, G\in\mathbf{L}_{\mathrm{HC}}, w\in\mathbf{R}_{\mathrm{HC}}(G), r\in\{0,1\}^{3n}}$
- $\left\{\mathsf{Exp}_1^{\mathrm{AH}}(n, G, w, r)\right\}_{n\in\mathbb{N}, G\in\mathbf{L}_{\mathrm{HC}}, w\in\mathbf{R}_{\mathrm{HC}}(G), r\in\{0,1\}^{3n}}$

4.3.5 Barak's Non-black-box Zero-knowledge Protocols

In this section, we recall Barak's non-black-box zero-knowledge protocol [Bar01]. As explained in Section 4.2, in our LRZK protocol we use a variant of so called "encrypted" Barak's preamble [PR05b, PR05a], which is based on the preamble stage of Barak's non-black-box zero-knowledge protocol.

Barak's non-black-box zero-knowledge protocol is constructed from any collision-resilient hash function family $\mathcal{H}$. Informally speaking, Barak's protocol $\mathsf{BarakZK}$ proceeds as follows.

$\underline{\text{Protocol } \mathsf{BarakZK}}$

1. The verifier V sends a random hash function $h \in \mathcal{H}$ and the first-round message $r_1 \in \{0, 1\}^{3n}$ of Naor's commitment scheme Com to the prover P.
2. P sends $c \leftarrow \mathsf{Com}_{r_1}(0^n)$ to V. Then, V sends random string r_2 to P.
3. P proves the following statement by a witness-indistinguishable argument.
   - $x \in L$, or
   - $(h, c, r_2) \in \Lambda$, where $(h, c, r_2) \in \Lambda$ holds if and only if there exists a machine Π such that c is a commitment to $h(\Pi)$ and Π outputs r_2 in $n^{\log\log n}$ steps.

Note that the statement proven in the last step is not in $\mathcal{NP}$. Thus, P proves this statement by a witness-indistinguishable *universal argument* (WIUA), with which P can prove any statement in $\mathcal{NEXP}$ (cf. Section 2.5). Intuitively, $\mathsf{BarakZK}$ is sound since $\Pi(c) \neq r$ holds with overwhelming probability even when a cheating prover P^* commits to $h(\Pi)$ for a machine Π. On the other hand, the zero-knowledge property can be proven by using a simulator that commits to $h(\Pi)$ such that Π is a machine that emulates the cheating verifier V^*; since $\Pi(c) = V^*(c) = r$ holds from the definition, the simulator can give a valid proof in the last step.

For our purpose, it is convenient to consider a variant of $\mathsf{BarakZK}$ that we denote by $\langle P_{\mathrm{B}}, V_{\mathrm{B}}\rangle$. $\langle P_{\mathrm{B}}, V_{\mathrm{B}}\rangle$ is the same as $\mathsf{BarakZK}$ except that in the last step, instead of proving

$x \in L \vee (h, c, r_2) \in \Lambda$ by using WIUA, P proves $(h, c, r_2) \in \Lambda$ by using a four-round public-coin universal argument system UA [BG08]. (Hence, $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ is no longer zero-knowledge protocol.) The formal description of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ is shown in Figure 4.1. We remark that in $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$, the language proven in the last step is replaced with a slightly more complex language as in, e.g., [Bar01, PR05b, PR05a, Pan14]. This replacement is important for using $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ in the setting of leakage-resilient zero-knowledge, because the cheating verifier can obtain arbitrary information (i.e., leakage) before sending r_2.

Stage 1:

The verifier V_{B} sends a random hash function $h \in \mathcal{H}$ to the prover P_{B}, where the domain of h is $\{0,1\}^*$ and the range of h is $\{0,1\}^n$. V_{B} also sends $r_1 \in \{0,1\}^{3n}$ (the first-round message of Com) to P_{B}.

Stage 2:

1. P_{B} computes $c \leftarrow \mathsf{Com}_{r_1}(0^n)$ and send c to V_{B}.
2. V_{B} sends random $r_2 \in \{0,1\}^{n+n^2}$ to P_{B}.

Stage 3: P_{B} proves statement $(h, r_1, c, r_2) \in \Lambda$ by using UA.

1. V_{B} sends the first-round message α.
2. P_{B} sends the second-round message β.
3. V_{B} sends the third-round message γ.
4. P_{B} sends the fourth-round message δ.

. . . . . .

Language Λ:

$(h, r_1, c, r_2) \in \Lambda$ if and only if there exist

- a machine Π
- randomness rand for Com
- a string y such that $|y| \leq n^2$

such that

- $c = \mathsf{Com}_{r_1}(h(\Pi); \mathsf{rand})$, and
- $\Pi(c, y)$ outputs r_2 within $n^{\log \log n}$ steps.

Figure 4.1: Encrypted Barak's preamble $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$.

In essentially the same way as the soundness of $\mathsf{BarakZK}$, we can prove the following lemma on $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$, which roughly states that there exists a "hard" language $\mathbf{L}_{\mathrm{B}}$ on the transcript of $\langle P_{\mathrm{B}}, V_{\mathrm{B}} \rangle$ such that no cheating prover can generate a transcript that is included in $\mathbf{L}_{\mathrm{B}}$.

Lemma 4.1 (Soundness). *Let $\boldsymbol{L}_{\mathrm{B}}$ be the language defined in Figure 4.2. Then, for any cheating prover P^* against $\langle P_{\mathrm{B}}, V_{\mathrm{B}}\rangle$, any $n \in \mathbb{N}$, and any $z \in \{0,1\}^*$,*

$$\Pr\left[\tau \in \boldsymbol{L}_{\mathrm{B}} \;\middle|\; \tau \leftarrow \mathsf{trans}\left[P^*(1^n, z) \leftrightarrow V_{\mathrm{B}}(1^n)\right]\right] \leq \mathsf{negl}(n) \enspace .$$

Language $\mathbf{L}_{\mathrm{B}}$:
$\tau = (h, r_1, c, r_2, \alpha, \beta, \gamma, \delta) \in \mathbf{L}_{\mathrm{B}}$ if and only if $(\alpha, \beta, \gamma, \delta)$ is an accepting transcript of UA for statement $(h, r_1, c, r_2) \in \Lambda$.

Figure 4.2: A "hard" language $\mathbf{L}_{\mathrm{B}}$.

Proof sketch of Lemma 4.1. We first remark that the language Λ depicted in Figure 4.1 is overly simplified and therefore we can prove this lemma only when the underlying hash function family $\mathcal{H}$ is secure against $\mathsf{poly}(n^{\log\log n})$-time adversaries. By using the language given in [BG08], we can prove this lemma even when $\mathcal{H}$ is secure only against polynomial-time adversaries.

Assume for contradiction that there exists P^* such that for infinitely many n's, there exists $z \in \{0,1\}^*$ such that the following holds for a polynomial $p(\cdot)$.

$$\Pr\left[\tau \in \mathbf{L}_{\mathrm{B}} \;\middle|\; \tau \leftarrow \mathsf{trans}\left[P^*(1^n, z) \leftrightarrow V_{\mathrm{B}}(1^n)\right]\right] \geq \frac{1}{p(n)} \enspace .$$

Fix any such P^*, n, and z. Then, consider interacting with P^* in the following way.

1. Interacts with P^* as an honest V_{B} until the end of $\langle P_{\mathrm{B}}, V_{\mathrm{B}}\rangle$. Let (h, r_1, c, r_2) be the transcript of the first two stages. If the UA proof in the last stage in not accepting, abort. Otherwise, extracts witness $w = (\Pi, R, y)$ for $(h, r_1, c, r_2) \in \Lambda$ using the global extractability of UA. (From the definition of the global extractability of UA, this extraction takes at most $\mathsf{poly}(n^{\log\log n})$ steps.)

2. Rewind P^* to the point just before sending r_2 to P^*, and interacts with P^* again as an honest V_{B} with fresh randomness until the end of $\langle P_{\mathrm{B}}, V_{\mathrm{B}}\rangle$. Let (h, r_1, c, r_2') be the transcript of the first two stages. If the UA proof in not accepting, abort. Otherwise, extracts witness $w' = (\Pi', R', y')$ for $(h, r_1, c, r_2') \in \Lambda$ using the extractability of UA.

From an average argument and the extractability of UA, we can obtain w and w' with probability $1/p'(n)$ for a polynomial $p'(\cdot)$. We then show that when we obtain w and w', we can obtain a collision of h. First, observe that since Π is deterministic, we have

$$\left|\left\{r : \exists y \in \{0,1\}^* \text{ s.t. } |y| \leq n^2 \wedge \Pi(c, y) = r\right\}\right| \leq 2^{n^2+1} \enspace .$$

Since r_2' is chosen uniformly at random from $\{0,1\}^{n+n^2}$, the probability that there exists $y \in \{0,1\}^{n^2}$ such that $\Pi(c, y) = r_2'$ is at most $2^{n^2+1}/2^{n+n^2} = 1/2^{n-1}$. Then, since we have $\Pi'(c, y_2') = r_2'$ because w' is a valid witness, we have $\Pi \neq \Pi'$ except with probability $1/2^{n-1}$. Furthermore, since both $h(\Pi)$ and $h(\Pi')$ are the committed value of c, from the statistical binding property of Com, $h(\Pi) = h(\Pi')$ holds except with negligible probability. Hence, the pair of Π and Π' is a collision of h except with negligible probability. □

4.3.6 Somewhat Extractable Commitment Schemes

In this section, we introduce a commitment scheme that satisfies only very weak extractability that we call *somewhat extractability*. This scheme will be used in our variant of encrypted Barak's preamble in Section 4.4.1. As mentioned in Remark 4.1 in Section 4.2.2, an important point on this scheme is that the committer sends only pseudorandom messages while it can be constructed from one-way functions.

Concretely, we consider the commitment scheme SWExtCom in Figure 4.3. SWExtCom is the same as the extractable commitment scheme of [PW09] except that in the last step, the committer simply reveals the values that it committed to in the first step (instead of decommitting the commitments). Because of this simplification, SWExtCom does not satisfy extractability in the standard sense. Still, it is not hard to see that SWExtCom satisfies extractability in the sense that, given two valid commitments c and c' such that the transcripts of the commit stage are identical but those of the challenge stage are different, then the committed value of c can be extracted. Formally, SWExtCom satisfies the following extractability.

Lemma 4.2 (Somewhat extractability)**.** *Let us say that two commitments*

$$c = (\{c_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e_i\}_{i\in[n]}, \{a_{i,e_i}\}_{i\in[n]}) \quad \textit{and} \quad c' = (\{c'_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e'_i\}_{i\in[n]}, \{a'_{i,e_i}\}_{i\in[n]})$$

are ***admissible*** *if*

- $c_{i,b} = c'_{i,b}$ *for every* $i \in [n]$ *and* $b \in \{0,1\}$,
- *there exists* $i^* \in [n]$ *such that* $e_{i^*} \neq e'_{i^*}$, *and*
- *the committed value of* $c_{i,b}$ *is uniquely determined for every* $i \in [n]$ *and* $b \in \{0,1\}$.

Let $\mathsf{Extract}(\cdot,\cdot)$ *be the algorithm shown in Figure 4.3. Then, for any two admissible commitments* c *and* c'*, if both* c *and* c' *are valid,* $\widetilde{v} \stackrel{\text{def}}{=} \mathsf{Extract}(c, c')$ *is equal to* $\mathsf{Value}(c)$ *(i.e.,* $\widetilde{v}$ *is the committed value of* c*).*

Proof. First, when c and c' are valid, $a_{i^*,e_{i^*}}$ and $a'_{i^*,e'_{i^*}}$ are the committed values of $c_{i^*,e_{i^*}}$ and $c_{i^*,e'_{i^*}}$ (since otherwise, any decommitments of c and c' would be rejected because the decommitted values of $c_{i^*,e_{i^*}}$ and $c_{i^*,e'_{i^*}}$ are not consistent with $a_{i^*,e_{i^*}}$ and $a'_{i^*,e'_{i^*}}$). Second, when c and c' are valid, the committed value of c can be computed by XORing the committed values of $c_{i^*,e_{i^*}}$ and $c_{i^*,e'_{i^*}}$ (since otherwise, any decommitments of c and c' would be rejected). From these, the lemma follows. □

A nice property of SWExtCom is that all the messages that the committer sends in the commit phase are pseudorandom. Formally, we have the following lemma.

Lemma 4.3 (Existence of public-coin fake committing algorithm)**.** *Let* C *be an honest committer algorithm of* SWExtCom*. There exists a* PPT *public-coin algorithm* C_{pub} *such that for any* PPT *cheating receiver* R^* *that interacts with* C *in the commit phase of* SWExtCom*, the following ensembles are computationally indistinguishable.*

- $\{\mathsf{output}_{R^*}[C(v) \leftrightarrow R^*(1^n, z)]\}_{n\in\mathbb{N},v\in\{0,1\}^n,z\in\{0,1\}^*}$

Commit phase. The committer C and the receiver R receive common input 1^n. To commit to $v \in \{0,1\}^n$, the committer C does the following with the receiver R.

Commit stage.
For each $i \in [n]$, the committer C chooses a pair of random n-bit strings $(a_{i,0}, a_{i,1})$ such that $a_{i,0} \oplus a_{i,1} = v$. Then, for each $i \in [n]$ in parallel, C commits to $a_{i,0}$ and $a_{i,1}$ by using Naor's commitment scheme Com. For each $i \in [n]$ and $b \in \{0,1\}$, let $c_{i,b}$ be the commitment to $a_{i,b}$.

Challenge stage.
R sends random n-bit string $e = (e_1, \ldots, e_n)$ to C.

Reply stage.
For each $i \in [n]$, C sends a_{i,e_i} to R.
COMMENT: *C just sends a_{i,e_i} and does not decommit c_{i,e_i}.*

Decommit phase. C sends v to R and decommits $c_{i,b}$ to $a_{i,b}$ for all $i \in [n]$ and $b \in \{0,1\}$. R checks whether $a_{1,0} \oplus a_{1,1} = \cdots = a_{n,0} \oplus a_{n,1} = v$ holds and whether $a_{1,e_1}, \ldots, a_{n,e_n}$ are equal to the values that were revealed in the commit phase.

. . . . . .

Extracting algorithm $\mathsf{Extract}$**.**
On input two commitments $c = (\{c_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e_i\}_{i\in[n]}, \{a_{i,e_i}\}_{i\in[n]})$ and $c' = (\{c'_{i,b}\}_{i\in[n],b\in\{0,1\}}, \{e'_i\}_{i\in[n]}, \{a'_{i,e_i}\}_{i\in[n]})$ such that $c_{i,b} = c'_{i,b}$ for every $i \in [n]$ and $b \in \{0,1\}$, do the following.

1. Find any $i \in [n]$ such that $e_i \neq e'_i$. If no such i exist, output fail.
2. Output $\widetilde{v} \stackrel{\text{def}}{=} a_{i,e_i} \oplus a'_{i,e'_i}$.

Figure 4.3: A somewhat extractable commitment scheme $\mathsf{SWExtCom}$

- $\left\{\mathsf{output}_{R^*}\left[C_{\mathrm{pub}}(1^n) \leftrightarrow R^*(1^n, z)\right]\right\}_{n\in\mathbb{N},v\in\{0,1\}^n,z\in\{0,1\}^*}$

Proof sketch. C_{pub} is an algorithm that is the same as C except that, instead of sending commitments of Com, it sends fake commitments of Com using $\mathsf{Com}_{\mathrm{pub}}$ (i.e., sends random strings with the same length as the Com commitments; cf. Section 4.3.1). Since Com has pseudorandom range, the indistinguishability can be proven by using a standard hybrid argument (in which the commitments of Com are replaced with random strings one by one). The formal proof is omitted. □

4.4 Building Blocks

In this section, we introduce two building blocks that we use in our LRZK protocol.

4.4.1 Special-purpose Encrypted Barak's Preamble

In our LRZK protocol, we use a variant of so called "encrypted" Barak's preamble [PR05b, PR05a]. The encrypted Barak's preamble is the same as (a variant of) Barak's non-black-box zero-knowledge protocol $\langle P_B, V_B \rangle$ in Section 4.3.5 except that P_B commits to its UA messages β and δ instead of sending them in clear. In our variant of the encrypted Barak's preamble, instead of giving valid commitments, P_B gives fake commitments of Com and SWExtCom by using $\mathsf{Com}_{\mathrm{pub}}$ and C_{pub} (cf. Sections 4.3.1 and 4.3.6). A nice property of our variant is that the prover sends only random strings; as will become clear later, this property is useful for constructing leakage-resilient protocols. The formal description of our variant, which we denote by $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$, is shown in Figure 4.4.

We first show that, as in the case of $\langle P_B, V_B \rangle$, there exists a "hard" language on the transcript of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$.

Lemma 4.4 (Soundness)**.** *Let $\mathbb{L}_B$ be the language defined in Figure 4.5. Then, for any cheating prover $\mathbb{P}^*$ against $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$, any $n \in \mathbb{N}$, and any $z \in \{0,1\}^*$,*

$$\Pr\left[\tau \in \mathbb{L}_B \;\middle|\; \tau \leftarrow \mathsf{trans}\left[\mathbb{P}^*(1^n, z) \leftrightarrow \mathbb{V}_B(1^n)\right]\right] \leq \mathsf{negl}(n) \ .$$

Proof. Assume for contradiction that there exists $\mathbb{P}^*$ such that for infinitely many n's, there exists $z \in \{0,1\}^*$ such that

$$\Pr\left[\tau \in \mathbb{L}_B \;\middle|\; \tau \leftarrow \mathsf{trans}\left[\mathbb{P}^*(1^n, z) \leftrightarrow \mathbb{V}_B(1^n)\right]\right] \geq \frac{1}{p(n)}$$

for a polynomial $p(\cdot)$. We use $\mathbb{P}^*$ to construct a cheating prover P^* against $\langle P_B, V_B \rangle$ and show that it contradicts the soundness of $\langle P_B, V_B \rangle$ (i.e., Lemma 4.1).

Consider the following cheating prover P^* against $\langle P_B, V_B \rangle$. First, P^* internally invokes $\mathbb{P}^*$. Then, while externally interacting with an honest V_B of $\langle P_B, V_B \rangle$, P^* interacts with internal $\mathbb{P}^*$ as a verifier of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ in the following way.

- In Stage 1 and 2 (of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$), P^* forwards all messages from external V_B to internal $\mathbb{P}^*$ and forwards all messages from internal $\mathbb{P}^*$ to external V_B. (Notice that the verifier of $\langle P_B, V_B \rangle$ and that of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ are identical.) Let (h, r_1, c, r_2) be the transcript of these stages.
- In Stage 3-1, P^* forwards α from external V_B to internal $\mathbb{P}^*$.
- In Stage 3-2, P^* interacts with internal $\mathbb{P}^*$ as an honest receiver of SWExtCom and obtains $\widehat{\beta}_1$. Let st be the current state of $\mathbb{P}^*$. Then, P^* rewinds $\mathbb{P}^*$ to the point just before the challenge stage of SWExtCom, interacts with $\mathbb{P}^*$ again, and obtains $\widehat{\beta}_2$. Then, P^* computes a potential committed value $\widetilde{\beta} \stackrel{\mathrm{def}}{=} \mathsf{Extract}(\widehat{\beta}_1, \widehat{\beta}_2)$ of $\widehat{\beta}_1$ (recall that Extract is the extracting algorithm of SWExtCom shown in Figure 4.3) and sends $\widetilde{\beta}$ to external V_B.
- In Stage 3-3, P^* receives γ from V_B and sends it to internal $\mathbb{P}^*$ (which is restarted from state st).

Stage 1:

The verifier $\mathbb{V}_B$ sends a random hash function $h \in \mathcal{H}$ to the prover $\mathbb{P}_B$. $\mathbb{V}_B$ also sends $r_1 \in \{0,1\}^{3n}$ (the first-round message of Naor's commitment scheme Com) to $\mathbb{P}_B$.

Stage 2:

1. $\mathbb{P}_B$ gives a fake commitment c of Com to $\mathbb{V}_B$ by running $c \leftarrow \mathsf{Com}_{\text{pub}}(1^n)$.
2. $\mathbb{V}_B$ sends random $r_2 \in \{0,1\}^{n+n^2}$ to $\mathbb{P}_B$.

Stage 3 (Encrypted UA):

1. $\mathbb{V}_B$ sends the first-round message α of UA for statement $(h, r_1, c, r_2) \in \Lambda$.
2. $\mathbb{P}_B$ gives a fake commitment of $\mathsf{SWExtCom}$ to $\mathbb{V}_B$ by running $C_{\text{pub}}(1^n)$. Let $\widehat{\beta}$ be the fake commitment (i.e., the transcript of this step).
3. $\mathbb{V}_B$ sends the third-round message γ of UA for statement $(h, r_1, c, r_2) \in \Lambda$.
4. $\mathbb{P}_B$ gives a fake commitment of $\mathsf{SWExtCom}$ to $\mathbb{V}_B$ by running $C_{\text{pub}}(1^n)$. Let $\widehat{\delta}$ be the fake commitment.

. . . . . .

Language Λ (same as the one in Figure 4.1):

$(h, r_1, c, r_2) \in \Lambda$ if and only if there exist

- a machine Π
- randomness rand for Com
- a string y such that $|y| \leq n^2$

such that

- $c = \mathsf{Com}_{r_1}(h(\Pi); \mathsf{rand})$, and
- $\Pi(c, y)$ outputs r_2 within $n^{\log\log n}$ steps.

Figure 4.4: Special-purpose encrypted Barak's preamble $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$.

Language $\mathbb{L}_{\mathrm{B}}$:

$(h, r_1, c, r_2, \alpha, \widehat{\beta}, \gamma, \widehat{\delta}) \in \mathbb{L}_{\mathrm{B}}$ if and only if there exist

- decommitments $d_1, d_2 \in \{0,1\}^{\mathsf{poly}(n)}$ for $\mathsf{SWExtCom}$
- the second-round and the fourth-round messages $\beta, \delta \in \{0,1\}^n$ of UA

such that

- d_1 is a valid decommitment of $\widehat{\beta}$ to β, and
- d_2 is a valid decommitment of $\widehat{\delta}$ to δ, and
- $(\alpha, \beta, \gamma, \delta)$ is an accepting transcript of UA for statement $(h, r_1, c, r_2) \in \Lambda$.

Figure 4.5: Language $\mathbb{L}_{\mathrm{B}}$.

- In Stage 3-4, P^* interacts with internal $\mathbb{P}^*$ as an honest receiver of $\mathsf{SWExtCom}$ and obtains $\widehat{\delta}_1$. Then, P^* rewinds $\mathbb{P}^*$ to the point just before the challenge stage of $\mathsf{SWExtCom}$, interacts with $\mathbb{P}^*$ again, and obtains $\widehat{\delta}_2$. Then, P^* computes $\widetilde{\delta} := \mathsf{Extract}(\widehat{\delta}_1, \widehat{\delta}_2)$ and sends $\widetilde{\delta}$ to external V_{B}.

Whenever internal $\mathbb{P}^*$ aborts, P^* also aborts.

Before analyzing the success probability of P^*, we first introduce some terminologies regarding the internally emulated interaction between $\mathbb{P}^*$ and $\mathbb{V}_{\mathrm{B}}$. Let $\tau = (h, r_1, c, r_2, \alpha, \widehat{\beta}_1, \gamma, \widehat{\delta}_1)$ be its transcript. Notice that since P^* emulates $\mathbb{V}_{\mathrm{B}}$ for internal $\mathbb{P}^*$ perfectly, we have $\tau \in \mathbb{L}_{\mathrm{B}}$ with probability at least $1/p(n)$.

- We say that a transcript τ_1 up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-2 is *good* if under the condition that τ_1 is a prefix of τ, the probability that $\tau \in \mathbb{L}_{\mathrm{B}}$ holds is at least $1/2p(n)$.

- We say that a transcript τ_2 up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-4 is *good* if (1) a prefix of τ_2 up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-2 is good and (2) under the condition that τ_2 is a prefix of τ, the probability that $\tau \in \mathbb{L}_{\mathrm{B}}$ holds is at least $1/4p(n)$.

We then analyze the success probability of P^* as follows. Let $\mathtt{GOOD}_1$ be the event that a prefix of τ up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-2 is good, and let $\mathtt{GOOD}_2$ be the event that a prefix of τ up until the commit stage of $\mathsf{SWExtCom}$ in Stage 3-4 is good. From an average argument, we have

$$\Pr[\mathtt{GOOD}_1] \geq \frac{1}{2p(n)} \qquad \text{and} \qquad \Pr[\mathtt{GOOD}_2 \mid \mathtt{GOOD}_1] \geq \frac{1}{4p(n)} \ .$$

Hence, we have

$$\Pr[\mathtt{GOOD}_2] = \Pr[\mathtt{GOOD}_1 \wedge \mathtt{GOOD}_2] \geq \frac{1}{8\,(p(n))^2} \ . \tag{4.1}$$

(The first equation holds since GOOD_1 occurs whenever GOOD_2 occurs.) Also, from the definition of GOOD_2, we have

$$\Pr[\tau \in \mathbb{L}_B \mid \mathsf{GOOD}_2] \geq \frac{1}{4p(n)} \ . \tag{4.2}$$

Hence, from Equation (4.1) and (4.2), we have

$$\Pr[\mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_B] = \Pr[\mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_B] \geq \frac{1}{32\,(p(n))^3} \ . \tag{4.3}$$

Next, we observe that when the transcript up until the commit stage of SWExtCom in Stage 3-2 is good, $\mathbb{P}^*$ gives a valid commitment of SWExtCom in Stage 3-2 with probability at least $1/2p(n)$, and similarly, when the transcript up until the commit stage of SWExtCom in Stage 3-4 is good, $\mathbb{P}^*$ gives a valid commitment of SWExtCom in Stage 3-4 with probability at least $1/4p(n)$. (This is because when the transcript is in $\mathbb{L}_B$, the SWExtCom commitments in Stage 3-2 and 3-4 are valid.) Hence, under the condition that $\mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_B$, the probability that both of $\widehat{\beta}_2$ and $\widehat{\delta}_2$ are valid is at least $1/8(p(n))^2$. Also, from the definition of $\mathbb{L}_B$, both of $\widehat{\beta}_1$ and $\widehat{\delta}_1$ are valid when $\tau \in \mathbb{L}_B$, and furthermore, $\widehat{\beta}_1$ and $\widehat{\beta}_2$ (resp, $\widehat{\delta}_1$ and $\widehat{\delta}_2$) are admissible except with negligible probability. Hence, from Lemma 4.2, for $\widetilde{\beta} = \mathsf{Extract}(\widehat{\beta}_1, \widehat{\beta}_2)$ and $\widetilde{\delta} = \mathsf{Extract}(\widehat{\delta}_1, \widehat{\delta}_2)$ we have

$$\begin{aligned} &\Pr\left[\widetilde{\beta} = \mathsf{Value}(\widehat{\beta}_1) \wedge \widetilde{\delta} = \mathsf{Value}(\widehat{\delta}_1) \mid \mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_B\right] \\ &\geq \frac{1}{8(p(n))^2} - \mathsf{negl}(n) \ . \end{aligned} \tag{4.4}$$

Hence, from Equation (4.3) and (4.4), we have

$$\begin{aligned} &\Pr\left[\mathsf{GOOD}_1 \wedge \mathsf{GOOD}_2 \wedge \tau \in \mathbb{L}_B \wedge \widetilde{\beta} = \mathsf{Value}(\widehat{\beta}_1) \wedge \widetilde{\delta} = \mathsf{Value}(\widehat{\delta}_1)\right] \\ &\geq \frac{1}{256(p(n))^5} - \mathsf{negl}(n) \ . \end{aligned}$$

Notice that from the definition of $\mathbb{L}_B$, when $\tau \in \mathbb{L}_B \wedge \widetilde{\beta} = \mathsf{Value}(\widehat{\beta}_1) \wedge \widetilde{\delta} = \mathsf{Value}(\widehat{\delta}_1)$, it holds that $(\alpha, \widetilde{\beta}, \gamma, \widetilde{\delta})$ is an accepting UA proof for $(h, r_1, c, r_2) \in \Lambda$. Hence, we have

$$\Pr\left[(h, r_1, c, r_2, \alpha, \widetilde{\beta}, \gamma, \widetilde{\delta}) \in \mathbf{L}_B\right] \geq \frac{1}{256(p(n))^5} - \mathsf{negl}(n) \ ,$$

which contradicts Lemma 4.1. □

We next note that a non-black-box simulator can simulate the transcript τ in such a way that $\tau \in \mathbb{L}_B$ holds, and the simulator can additionally output a witness for $\tau \in \mathbb{L}_B$.

Lemma 4.5 (Simulatability)**.** *Let $\mathbb{L}_B$ be the language defined in Figure 4.5. Then, for any PPT cheating verifier $\mathbb{V}^*$ against $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$, there exists a PPT simulator $\mathbb{S}$ such that the following hold.*

- *Let $\mathbb{S}_1(x, z)$ be the random variable representing the first output of $\mathbb{S}(x, z)$. Then, the following indistinguishability holds.*

$$\{\mathsf{view}_{\mathbb{V}^*}\,[\mathbb{P}_{\mathrm{B}}(1^n) \leftrightarrow \mathbb{V}^*(1^n, z)]\}_{n\in\mathbb{N},z\in\{0,1\}^*} \overset{c}{\approx} \{\mathbb{S}_1(1^n, z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$$

- *For any $n \in \mathbb{N}$ and $z \in \{0,1\}^*$, the following holds.*

$$\Pr\left[w \in \boldsymbol{R}_{\mathbb{L}_{\mathrm{B}}}(\tau) \;\middle|\; \begin{array}{l}(v, w) \leftarrow \mathbb{S}(1^n, z);\\ \textit{reconstruct transcript } \tau \textit{ from view } v \textit{ of } \mathbb{V}^*\end{array}\right] \geq 1 - \mathsf{negl}(n)$$

This lemma can be proven in essentially the same way as the zero-knowledge property of Barak's non-black-box zero-knowledge protocol. For completeness, we give a proof sketch below.

Proof sketch of Lemma 4.5. To simulate the view of $\mathbb{V}^*$, the simulator $\mathbb{S}$ internally invokes $\mathbb{V}^*$ and interacts with it as follows.

- After receiving h and r_1 in Stage 1, $\mathbb{S}$ sends $c \leftarrow \mathsf{Com}_{r_1}(h(\mathbb{V}^*))$ to $\mathbb{V}^*$ in Stage 2-1. Let rand be the randomness that was used in this step.
- After receiving r_2 in Stage 2-2 and α in Stage 3-1, $\mathbb{S}$ computes the second-round UA message β by using witness $(\mathbb{V}^*, \mathsf{rand}, \varepsilon)$ for $(h, r_1, c, r_2) \in \Lambda$ (where ε is an empty string) and then honestly commits to β by using $\mathsf{SWExtCom}$. Let $\widehat{\beta}$ be the commitment and d_1 be the decommitment.
- After receiving γ in Stage 3-3, $\mathbb{S}$ computes the fourth-round UA message δ and then honestly commits to δ by using $\mathsf{SWExtCom}$. Let $\widehat{\delta}$ be the commitment and d_2 be the decommitment.

$\mathbb{S}$ then outputs (v, w), where v is the view of internal $\mathbb{V}^*$ and $w \overset{\text{def}}{=} (d_1, d_2, \beta, \delta)$. Let $\tau := (h, r_1, c, r_2, \alpha, \widehat{\beta}, \gamma, \widehat{\delta})$.

We analyze $\mathbb{S}$ as follows. First, from the hiding property of Com and the indistinguishability of C_{pub} (Lemma 4.3), v is indistinguishable from the real view of $\mathbb{V}^*$. Next, from the definitions of Λ and $\mathbb{L}_{\mathrm{B}}$, we have $\tau \in \mathbb{L}_{\mathrm{B}}$ and w is its witness. Hence, the lemma follows. □

4.4.2 Special-purpose Instance-dependent Commitment

In our LRZK protocol, we use a special-purpose instance-dependent commitment scheme $\mathsf{GJS\text{-}Com}$, which is shown in Figure 4.6. $\mathsf{GJS\text{-}Com}$ is parametrized by two graphs, G and G', and obtained by modifying Hamiltonicity commitment scheme $\mathsf{H\text{-}Com}_{G,r}$ (Section 4.3.3) in such a way that the adjacency matrix is committed to by using $\mathsf{AH\text{-}Com}_{G',r}$. $\mathsf{GJS\text{-}Com}$ inherits many properties from $\mathsf{H\text{-}Com}$—hiding, binding, and equivocality—and additionally, thanks to the adaptive security of $\mathsf{AH\text{-}Com}$, it provides adaptive security in the following sense: When $G \in \mathbf{L}_{\mathrm{HC}}$ and $G' \in \mathbf{L}_{\mathrm{HC}}$, a commitment to 1 can be explained as a valid commitment to 0, and furthermore, even after being explained as a commitment to 0, it can be decommitted to 1 in a consistent way. Details follow.

Parameters:

- Security parameter n.
- Two graphs G and G', where the number of vertices in G is $q = \mathsf{poly}(n)$ and that in G' is $q' = \mathsf{poly}'(n)$.

Inputs:

- C has secret input $b \in \{0,1\}$, which is the value to be committed to.

Commit phase:

1. R sends the first-round message $r \in \{0,1\}^{3n}$ of Com.
2. **To commit to 0,** C chooses a random permutation π over the vertices of G, computes $H_0 := \pi(G)$, and commits to its adjacency matrix $A_0 = \{a_{0,i,j}\}_{i,j\in[q]}$ by using $\mathsf{AH\text{-}Com}_{G',r}$, i.e., sends $c_{i,j} \leftarrow \mathsf{AH\text{-}Com}_{G',r}(a_{0,i,j})$ for every $i, j \in [q]$.

   To commit to 1, C chooses a random q-cycle graph H_1 and commits to its adjacency matrix $A_1 = \{a_{1,i,j}\}_{i,j\in[q]}$ by using $\mathsf{AH\text{-}Com}_{G',r}$, i.e., sends $c_{i,j} \leftarrow \mathsf{AH\text{-}Com}_{G',r}(a_{1,i,j})$ for every $i, j \in [q]$.

Let $\mathsf{GJS\text{-}Com}_{G,G',r}(\cdot)$ be a function that, on input $b \in \{0,1\}$, computes a commitment to b as above by considering r as the first-round message from the receiver.

Decommit phase:

- **When** C **committed to 0,** it reveals π and decommits $c_{i,j}$ to $a_{0,i,j}$ for every $i, j \in [q]$. R verifies whether the decommitted matrix is the adjacency matrix of $\pi(G)$.
- **When** C **committed to** 1, it decommits $c_{i,j}$ to 1 for every i, j such that edge $(i.j)$ is on the q-cycle in H_1 (i.e., every i, j such that $a_{1,i,j} = 1$). R verifies whether the decommitted entries correspond to the edges on a Hamiltonian cycle.

Let $\mathsf{GJS\text{-}Dec}_r(\cdot)$ be a function that, on input (c, b, ρ) such that $\mathsf{GJS\text{-}Com}_{G,G',r}(b;\rho) = c$, outputs a decommitment to b as above.

Figure 4.6: Special-purpose instance-dependent commitment GJS-Com.

Lemma 4.6 (Hiding and binding)**.** GJS-Com *is computationally hiding. Furthermore, it is statistically binding when* $G \notin \boldsymbol{L}_{\mathrm{HC}}$ *and* $G' \notin \boldsymbol{L}_{\mathrm{HC}}$.

Proof. The hiding property follows directly from the hiding property of $\mathsf{AH\text{-}Com}_{G'}$. To see the binding property, observe the following: When $G' \notin \mathbf{L}_{\mathrm{HC}}$, $\mathsf{AH\text{-}Com}_{G'}$ is statistically binding and therefore the matrix committed to in the commit phase of GJS-Com is uniquely determined except with negligible probability; furthermore, when the com-

mitted matrix is uniquely determined and $G \notin \mathbf{L}_{\mathrm{HC}}$, decommitting to both 0 and 1 is clearly impossible; hence, when $G \notin \mathbf{L}_{\mathrm{HC}}$ and $G' \notin \mathbf{L}_{\mathrm{HC}}$, a commitment of GJS-Com can be decommitted to both 0 and 1 only with negligible probability. □

Lemma 4.7 (Equivocality)**.** *There exists an algorithm* GJS-EquivToOne *that is parametrized by graphs G, G' and a string $r \in \{0,1\}^{3n}$ and satisfies the following: When $G \in \boldsymbol{L}_{\mathrm{HC}}$, on input any $w \in \boldsymbol{R}_{\mathrm{HC}}(G)$ and any c and ρ such that* $\mathsf{GJS\text{-}Com}_{G,G',r}(0;\rho) = c$, $\mathsf{GJS\text{-}EquivToOne}_{G,G',r}$ *outputs a valid decommitment of c to* 1.

Proof. We need to show that, on inputs a commitment c to 0, a witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$, and randomness ρ that is used to compute c, an algorithm GJS-EquivToOne can decommit c to 1.

GJS-EquivToOne decommits c to 1 as follows. From the construction of GJS-Com, commitment c consists of $\{c_{i,j}\}_{i,j\in[q]}$, which are AH-Com commitments to the adjacency matrix of $H_0 = \pi(G)$. To decommit c to 1, GJS-EquivToOne need to decommit some of $\{c_{i,j}\}_{i,j\in[q]}$ to 1 so that the decommitted entries of the matrix correspond to the edges on a Hamiltonian cycle in a q-vertex graph. To do such decommitments, GJS-EquivToOne first computes a Hamiltonian cycle $\pi(w)$ in H_0 by using Hamiltonian cycle w in G and permutation π (which is included in ρ). Then, GJS-EquivToOne decommits $c_{i,j}$ to $a_{i,j}$ honestly for every i, j such that (i, j) is an edge on $\pi(w)$. Clearly, this is a valid decommitment to 1. □

Lemma 4.8 (Adaptive security)**.** *There exists an algorithm* GJS-ExplainAsZero *that is parametrized by graphs G, G' and a string $r \in \{0,1\}^{3n}$ and satisfies the following.*

Correctness. *When $G, G' \in \boldsymbol{L}_{\mathrm{HC}}$, on input any $w \in \boldsymbol{R}_{\mathrm{HC}}(G)$ and $w' \in \boldsymbol{R}_{\mathrm{HC}}(G')$ and any c and ρ_1 such that* $\mathsf{GJS\text{-}Com}_{G,G',r}(1;\rho_1) = c$, $\mathsf{GJS\text{-}ExplainAsZero}_{G,G',r}$ *outputs ρ_0 such that* $\mathsf{GJS\text{-}Com}_{G,G',r}(0;\rho_0) = c$.

Indistinguishability. *For security parameter $n \in \mathbb{N}$, graphs $G, G' \in \boldsymbol{L}_{\mathrm{HC}}$, witnesses $w \in \boldsymbol{R}_{\mathrm{HC}}(G)$ and $w' \in \boldsymbol{R}_{\mathrm{HC}}(G')$, and string $r \in \{0,1\}^{3n}$, consider the following two probabilistic experiments.*

$\underline{Exp_0^{\mathrm{GJS}}(n, G, G', w, w', r)}$

`/* commit to 0 and decommit it to 1 using equivocality */`

1. *Compute* $c \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(0)$.
   Let ρ_0 be the randomness used in GJS-Com.
2. *Compute* $d_1 := \mathsf{GJS\text{-}EquivToOne}_{G,G',r}(c, w, \rho_0)$.
3. *Output* (c, ρ_0, d_1).

$\underline{Exp_1^{\mathrm{GJS}}(n, G, G', w, w', r)}$

`/* commit & decommit to 1 and explain it as commitment to 0 */`

1. *Compute* $c \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(1)$.
   Let ρ_1 be the randomness used in GJS-Com.
   Compute $d_1 := \mathsf{GJS\text{-}Dec}_{G,G',r}(c, 1, \rho)$.
2. *Compute* $\rho_0 := \mathsf{GJS\text{-}ExplainAsZero}_{G,G',r}(c, w, w', \rho_1)$.

3. *Output* (c, ρ_0, d_1).

Let $\mathsf{Exp}_b^{\mathrm{GJS}}(n, G, G', w, w', r)$ *be the random variable representing the output of* $\mathit{Exp}_b^{\mathrm{GJS}}(n, G, G', w, w', r)$ *for each* $b \in \{0, 1\}$. *Then, the following two ensembles are computationally indistinguishable.*

- $\left\{\mathsf{Exp}_0^{\mathrm{GJS}}(n, G, G', w, w', r)\right\}_{n\in\mathbb{N}, G, G'\in \boldsymbol{L}_{\mathrm{HC}}, w\in \boldsymbol{R}_{\mathrm{HC}}(G), w'\in \boldsymbol{R}_{\mathrm{HC}}(G'), r\in\{0,1\}^{3n}}$
- $\left\{\mathsf{Exp}_1^{\mathrm{GJS}}(n, G, G', w, w', r)\right\}_{n\in\mathbb{N}, G, G'\in \boldsymbol{L}_{\mathrm{HC}}, w\in \boldsymbol{R}_{\mathrm{HC}}(G), w'\in \boldsymbol{R}_{\mathrm{HC}}(G'), r\in\{0,1\}^{3n}}$

Proof. GJS-ExplainAsZero is shown in Figure 4.7. A key idea behind this construction is that given the ability to explain AH-Com commitments to 0 as AH-Com commitments to 1, we can explain a commitment to 1 (which is AH-Com commitments to the adjacency matrix of a cycle graph) as a commitment to 0 (which is AH-Com commitments to the adjacency matrix of a Hamiltonian graph G). Intuitively, this is because a cycle graph can be transformed to any Hamiltonian graph by appropriately adding edges (which corresponds to changing some entries of the adjacency matrix from 0 to 1).

Parameter:

- Graphs $G, G' \in \mathbf{L}_{\mathrm{HC}}$
- String $r \in \{0, 1\}^{3n}$

Input:

- Witnesses $w \in \mathbf{R}_{\mathrm{HC}}(G)$ and $w' \in \mathbf{R}_{\mathrm{HC}}(G')$
- Commitment c and randomness ρ_1 s.t. $\mathsf{GJS\text{-}Com}_{G,G',r}(1; \rho_1) = c$

Output:

1. Parse c as $\{c_{i,j}\}_{i,j\in[q]}$, where each $c_{i,j}$ is a AH-Com commitment. Also, from ρ_1, reconstruct $A_1 = \{a_{1,i,j}\}_{i,j\in[q]}$ and $\{\sigma_{1,i,j}\}_{i,j\in[q]}$ such that A_1 is the adjacency matrix of a q-cycle graph H_1 and $\mathsf{AH\text{-}Com}_{G',r}(a_{1,i,j}; \sigma_{1,i,j}) = c_{i,j}$ for every $i, j \in [q]$.
2. Choose a random permutation π under the condition that a q-cycle in $H_0 \stackrel{\mathrm{def}}{=} \pi(G)$ coincides with the q-cycle in H_1 (i.e., H_0 has the same cycle as H_1).[a] Let $A_0 = \{a_{0,i,j}\}_{i,j\in[q]}$ be the adjacency matrix of H_0.
3. For every $i, j \in [q]$, define $\sigma_{0,i,j}$ by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \sigma_{1,i,j}$ when $a_{0,i,j} = a_{1,i,j}$ and by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \mathsf{AH\text{-}ExplainAsOne}_{G',r}(w', c_{i,j}, \sigma_{1,i,j})$ when $a_{0,i,j} \neq a_{1,i,j}$.[b]
4. Outputs $\rho_0 \stackrel{\mathrm{def}}{=} (\pi, \{\sigma_{0,i,j}\}_{i,j\in[q]})$.

[a] Given w, this can be done efficiently.
[b] When $a_{0,i,j} \neq a_{i,j}$, it holds that $a_{0,i,j} = 1$ and $a_{1,i,j} = 0$; see the proof.

Figure 4.7: GJS-ExplainAsZero.

We first prove the correctness. A key observation is that since H_0 is defined in such a way that H_0 has the same q-cycle as H_1, for every $i, j \in [q]$ we have only the following three cases regarding the values of $a_{0,i,j}$ and $a_{1,i,j}$.

Case 1. $a_{0,i,j} = 0$, $a_{1,i,j} = 0$

Case 2. $a_{0,i,j} = 1$, $a_{1,i,j} = 1$

Case 3. $a_{0,i,j} = 1$, $a_{1,i,j} = 0$

In particular, we do not have the case that $a_{0,i,j} = 0$ and $a_{1,i,j} = 1$ because when $a_{1,i,j} = 1$, edge (i, j) is on the q-cycle in H_1, and therefore edge (i, j) is also on a q-cycle in H_0 and thus $a_{0,i,j} = 1$. Then, since we have only these three cases, from the property of $\mathsf{AH\text{-}ExplainAsOne}$ we have $\mathsf{AH\text{-}Com}_{G',r}(a_{0,i,j}; \sigma_{0,i,j}) = c_{i,j}$ for every i, j such that $a_{0,i,j} \neq a_{1,i,j}$. Therefore, the output ρ_0 satisfies $\mathsf{GJS\text{-}Com}_{G,G',r}(1; \rho) = c$.

We next prove the indistinguishability. Toward this end, we consider the following hybrid experiments.

Hybrid $\mathbf{HYB}_0$ is the same as $\mathrm{Exp}_0^{\mathrm{GJS}}(n, G, G', w, w', r)$. Recall that in $\mathrm{Exp}_0^{\mathrm{GJS}}$, output (c, ρ_0, d_1) is computed as follows:

- Choose $\rho_0 = (\pi, \{\sigma_{0,i,j}\}_{i,j\in[q]})$, where π is a randomly chosen permutation and each $\sigma_{0,i,j}$ is randomly chosen randomness for $\mathsf{AH\text{-}Com}$.
- Compute $c = \{c_{i,j}\}_{i,j\in[q]}$ by $c_{i,j} := \mathsf{AH\text{-}Com}_{G',r}(a_{0,i,j}; \sigma_{0,i,j})$ for each $i, j \in [q]$, where $A_0 = \{a_{0,i,j}\}_{i,j\in[q]}$ is the adjacency matrix of $H_0 = \pi(G)$.
- Define $d_1 \stackrel{\mathrm{def}}{=} \{\sigma_{0,i,j}\}_{(i,j)\in\pi(w)}$, where $\pi(w)$ is the set of the edges on the Hamiltonian cycle in H_0 that is obtained by applying π on Hamiltonian cycle w in G.

Hybrid $\mathbf{HYB}_1$ is the same as HYB_0 except that π is chosen as follows:

1. Choose a random q-cycle graph H_1. Let $A_1 = \{a_{1,i,j}\}_{i,j\in[q]}$ be the adjacency matrix of H_1.
2. Choose a random permutation π under the condition that a q-cycle in $H_0 = \pi(G)$ coincides with the q-cycle in H_1.

Hybrid $\mathbf{HYB}_2$ is the same as HYB_1 except for the following.

- $c_{i,j}$ is computed by $c_{i,j} := \mathsf{AH\text{-}Com}_{G',r}(a_{1,i,j}; \sigma_{1,i,j})$ for every $i, j \in [q]$, where $\sigma_{1,i,j}$ is randomly chosen randomness.
- $\sigma_{0,i,j}$ is defined by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \sigma_{1,i,j}$ when $a_{0,i,j} = a_{1,i,j}$ and by $\sigma_{0,i,j} \stackrel{\mathrm{def}}{=} \mathsf{AH\text{-}ExplainAsOne}_{G',r}(w', c_{i,j}, \sigma_{1,i,j})$ when $a_{0,i,j} \neq a_{1,i,j}$.

Hybrid $\mathbf{HYB}_3$ is the same as $\mathrm{Exp}_1^{\mathrm{GJS}}(n, G, G', w, w', r)$.

From a hybrid argument, we can show the indistinguishability of $\mathrm{Exp}_0^{\mathrm{GJS}}$ and $\mathrm{Exp}_1^{\mathrm{GJS}}$ by showing the indistinguishability of each neighboring hybrids.

Claim 4.1. *The outputs of HYB_0 and HYB_1 are identically distributed.*

Proof. HYB_0 and HYB_1 differ only in the way π is chosen. However, the distribution of π is uniformly random in both hybrids. (In particular, the distribution of π is uniformly random in HYB_1 since H_1 is chosen randomly.) Hence, the claim follows. □

Claim 4.2. *The outputs of HYB_1 and HYB_2 are computationally indistinguishable.*

Proof. We first remark that, as noted above, we have $a_{0,i,j} = 1$ and $a_{1,i,j} = 0$ when $a_{0,i,j} \neq a_{1,i,j}$. Because of this, HYB_1 and HYB_2 differ only in that for every i, j such that $a_{0,i,j} \neq a_{1,i,j}$,

- in the case of HYB_1, $c_{i,j}$ is a commitment to 1 and $\sigma_{0,i,j}$ is randomly chosen randomness that is used to generate $c_{i,j}$, whereas
- in the case of HYB_2, $c_{i,j}$ is a commitment to 0 and $\sigma_{0,i,j}$ is the randomness that is computed by AH-ExplainAsOne.

Hence, the indistinguishability follows from the adaptive security of AH-Com. In particular, we can prove the indistinguishability by considering a sequence of intermediate hybrids $\mathrm{HYB}_{1,0}, \ldots, \mathrm{HYB}_{1,q^2}$ such that

- $\mathrm{HYB}_{1,0}$ is the same as HYB_1, and
- for every $u, v \in [q]$, $\mathrm{HYB}_{1,(u-1)q+v}$ is the same as $\mathrm{HYB}_{1,(u-1)q+v-1}$ except that $c_{u,v}$ and $\sigma_{0,u.v}$ are computed in the same way as in HYB_2,

and then proving the indistinguishability of each neighboring intermediate hybrids by designing an adversary against the adaptive security of AH-Com in a straight-forward manner so that, depending on the value of (c, ρ_1) that it receives externally, it internally emulates either $\mathrm{HYB}_{1,(u-1)q+v}$ or $\mathrm{HYB}_{1,(u-1)q+v-1}$ (i.e., when c is a commitment to 1 and ρ is its randomness, the adversary internally emulates $\mathrm{HYB}_{1,(u-1)q+v-1}$, and when c is a commitment to 0 and ρ is the randomness that is generated by AH-ExplainAsOne, the adversary internally emulates $\mathrm{HYB}_{1,(u-1)q+v}$). □

Claim 4.3. *The outputs of HYB_2 and HYB_3 are identically distributed.*

Proof. It can be seen by inspection that in HYB_2, the output (c, ρ_0, d_1) is computed in exactly the same way as in $\mathrm{Exp}_1^{\mathrm{GJS}}$. Hence, the claim follows. □

From these claims, we obtain the indistinguishability of $\mathrm{Exp}_0^{\mathrm{GJS}}$ and $\mathrm{Exp}_1^{\mathrm{GJS}}$. This concludes the proof of Lemma 4.8. □

4.5 Our Leakage-resilient Zero-knowledge Argument

In this section, by using the two building blocks $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ and GJS-Com in Section 4.4, we construct a constant-round LRZK argument system.

Theorem 4.1 (restatement of Main Theorem)**.** *Assume the existence of collision-resistant hash function family. Then, there exists a constant-round public-coin leakage-resilient zero-knowledge argument system* LR-ZK.

Proof. LR-ZK is shown in Figure 4.8. Since $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ can be constructed from any collision-resistant hash function family, and SWExtCom can be constructed from any one-way function (which can be obtained from any collision-resistant hash function family), LR-ZK can be constructed from any collision-resistant hash function family. Also, by inspection, it can be seen that LR-ZK is public-coin and has constant rounds.

In the following, we prove soundness in Section 4.5.1 and leakage-resilient zero-knowledgeness in Section 4.5.2.

4.5.1 Soundness

Lemma 4.9. LR-ZK *is sound against* PPT *adversaries.*

Proof. For any cheating PPT prover P^*, we show that P^* cannot give an accepting proof for a false statement $G \notin \mathbf{L}_{HC}$ except with negligible probability. Notice that from the soundness of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$, the statement τ generated in Stage 1 satisfies $\tau \notin \mathbb{L}_B$ except with negligible probability. Hence, it suffices to show that under the condition that $\tau \notin \mathbb{L}_B$ (and hence $G' \notin \mathbf{L}_{HC}$), P^* cannot give an accepting proof except with negligible probability.

A key observation is that when $G \notin \mathbf{L}_{HC}$ and $G' \notin \mathbf{L}_{HC}$, special-purpose instance-dependent commitment scheme $\mathsf{GJS\text{-}Com}_{G,G'}$ is statistically binding, and therefore the matrix that is committed to in Stage 2-1 is uniquely determined in each of the n iterations except with negligibly probability. Based on this observation, we can prove the soundness in essentially the same way as the soundness of Blum's Hamiltonicity protocol. Specifically, when the committed matrix is uniquely determined in each of the n iterations, P^* can give a valid response in Stage 2-3 with probability at most $1/2$ in each of n iterations. (This is because, when $G' \notin \mathbf{L}_{HC}$, no Hamiltonian graph is isomorphic to G'.) Hence, under the condition that $\tau \notin \mathbb{L}_B$, P^* can give an accepting proof with only negligible probability. This completes the proof of soundness. □

4.5.2 Leakage-resilient Zero-knowledgeness

Lemma 4.10. LR-ZK *is leakage-resilient zero-knowledge.*

In the following, we prove this lemma only w.r.t. a simplified version of LR-ZK in which Stage 2-1, 2-2, and 2-3 are executed only once (instead of executed n times in parallel). The proof w.r.t. the original version of LR-ZK can be obtained by modifying the following proof in a straightforward way.

Input. Common input to P and V is graph $G \in \mathbf{L}_{\mathrm{HC}}$. Let $n \stackrel{\mathrm{def}}{=} |G|$, and q be the number of vertices in G. Private input to P is witness $w \in \mathbf{R}_{\mathrm{HC}}(G)$.

Stage 1.

- P and V execute special-purpose encrypted Barak's preamble $\langle \mathbb{P}_{\mathrm{B}}, \mathbb{V}_{\mathrm{B}} \rangle$. Let τ be the transcript.
- P and V reduce statement "$\tau \in \mathbb{L}_{\mathrm{B}}$" to Hamiltonicity problem via general $\mathcal{NP}$ reduction. Let G' be the graph that P and V obtained. Let q' be the number of vertices in G'.

Stage 2.

- V sends the first-round message $r \in \{0,1\}^{3n}$ of Naor's commitment scheme Com to P.
- P and V do the following for n times in parallel.
  1. P commits to a $q' \times q'$ zero matrix in a bit-by-bit manner by using $\mathsf{GJS\text{-}Com}_{G,G',r}$. That is, P sends $c_{i,j} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(0)$ to V for every $i, j \in [q']$. Let $\rho_{i,j}$ be the randomness that was used to compute $c_{i,j}$.
  2. V sends a random bit $ch \in \{0,1\}$ to P.
  3. **When $ch = 0$:**
     - P chooses a random permutation π and computes $H_0 := \pi(G')$. Let $A_0 = \{a_{0,i,j}\}_{i,j \in [q']}$ be the adjacency matrix of H_0.
     - P sends π to V and decommits the $\mathsf{GJS\text{-}Com}$ commitments in Stage 2-1 to A_0 by using the equivocality of $\mathsf{GJS\text{-}Com}$. That is, for every $i, j \in [q]$, P sends an honest decommitment $d_{i,j} := \mathsf{GJS\text{-}Dec}_{G,G',r}(c_{i,j}, 0, \rho_{i,j})$ to V when $a_{0,i,j} = 0$ and sends a fake decommitment $d_{i,j} := \mathsf{GJS\text{-}EquivToOne}_{G,G',r}(c_{i,j}, w_0, \rho_{i,j})$ to V when $a_{0,i,j} = 1$.
     - V computes $H_0 = \pi(G')$ and verifies whether the decommitted matrix is equal to the adjacency matrix of H_0.

     When $ch = 1$:
     - P chooses a random q'-cycle graph H_1. Let $A_1 = \{a_{1,i,j}\}_{i,j \in [q']}$ be the adjacency matrix of H_1.
     - P decommits $c_{i,j}$ to $a_{1,i,j}$ for every i, j such that $a_{1,i,j} = 1$ (i.e., for every i, j such that edge (i, j) is on the q'-cycle of H_1). That is, for every such i and j, P sends a fake decommitment $d_{i,j} := \mathsf{GJS\text{-}EquivToOne}_{G,G',r}(c_{i,j}, w_0, \rho_{i,j})$ to V.
     - V checks whether the decommitted entries of the matrix correspond to the edges on a q'-cycle.

Figure 4.8: Constant-round leakage-resilient zero-knowledge argument $\mathsf{LR\text{-}ZK}$.

Proof. Without loss of generality, we assume that after receiving each message from the prover, the cheating verifier makes exactly a single leakage query. To see that we indeed do not lose generality, observe that instead of making two queries f_1 and f_2, the cheating verifier can always query a single query f such that, on input witness w and prover's randomness tape, it computes the first leakage $L_1 := f_1(w, \mathsf{tape})$, chooses the second query f_2 adaptively, computes the second leakage $L_2 := f_2(w, \mathsf{tape})$, and outputs (L_1, L_2).

In the following, we describe our simulator, observe that our simulator obtains the same amount of leakage as the adversary, and prove the indistinguishability of views.

4.5.2.1 Description of the simulator.

Given access to leakage oracle $\mathcal{L}_w$ and input (G, z), our simulator $\mathcal{S}$ simulates the view of cheating verifier V^* by internally invoking $V^*(G, z)$ and interacting with it as follows.

Simulating messages and leakages in Stage 1. Roughly speaking, $\mathcal{S}$ simulates the messages in Stage 1 by interacting with V^* in the same way as the simulator of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ (cf. Lemma 4.5). To simulate the leakages in Stage 1, $\mathcal{S}$ uses the fact that Stage 1 of $\mathsf{LR\text{-}ZK}$ is public coin w.r.t. the prover and therefore all the randomness that an honest prover generates during Stage 1 is the messages themselves. Specifically, $\mathcal{S}$ simulates the leakages by considering the messages msgs that it has sent to V^* thus far as the randomness of the prover. An issue is that due to the existence of leakage queries, $\mathcal{S}$ cannot use the simulator of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ in a modular way. Nonetheless, $\mathcal{S}$ can still use the technique used in the simulator of $\langle \mathbb{P}_B, \mathbb{V}_B \rangle$ as long as the length of the leakages is bounded by n^2. (Notice that when the length of leakage exceeds n^2, $\mathcal{S}$ can simply obtain a Hamiltonian cycle w of G from $\mathcal{L}_w$.)

Formally, $\mathcal{S}$ interacts with V^* as follows.

1. After receiving h and r_1 from V^*, $\mathcal{S}$ sends $c \leftarrow \mathsf{Com}_{r_1}(h(V^*))$ to V^*. Let rand be the randomness that was used in this step.
   Leakage query: When V^* makes a leakage query f, $\mathcal{S}$ does the following.
   - Let $\mathsf{tape} := c$.
   - If the output length of f is more than n^2, $\mathcal{S}$ obtains w from $\mathcal{L}_w$ and returns $f(w \| \mathsf{tape})$ to V^*.
   - Otherwise, $\mathcal{S}$ queries $f(\cdot, \mathsf{tape})$ to $\mathcal{L}_w$, obtains reply L from $\mathcal{L}_w$, and forwards L to V^*.

   If $\mathcal{S}$ obtained w, from now on $\mathcal{S}$ interacts with V^* in exactly the same way as an honest prover. Otherwise, do the following.

2. After receiving r_2 and α from V^*, $\mathcal{S}$ computes the second-round UA message β by using witness (V^*, rand, L) and then honestly commits to β by using $\mathsf{SWExtCom}$. Let $\widetilde{\beta}$ be the commitment and d_1 be the decommitment.
   Leakage query: When V^* makes a leakage query f, $\mathcal{S}$ sets $\mathsf{tape} := \mathsf{msgs}$, queries $f(\cdot, \mathsf{tape})$ to $\mathcal{L}_w$, and forwards the reply from $\mathcal{L}_w$ to V^*, where msgs are the messages that $\mathcal{S}$ has sent to V^* thus far.

3. After receiving γ from V^*, $\mathcal{S}$ computes the fourth-round UA message δ and then honestly commits to δ by using SWExtCom. Let $\widehat{\delta}$ be the commitment and d_2 be the decommitment.
   Leakage query: When V^* makes a leakage query f, $\mathcal{S}$ answers it in exactly the same way as above.

Let $\tau \stackrel{\text{def}}{=} (h, r_1, c, r_2, \alpha, \widehat{\beta}, \gamma, \widehat{\delta})$ and $\bar{w} \stackrel{\text{def}}{=} (d_1, d_2, \beta, \delta)$. Since (V^*, rand, L) is a valid witness for $(h, r_1, c, r_2) \in \Lambda$, we have $\tau \in \mathbb{L}_{\mathrm{B}}$ and $\bar{w} \in \mathbf{R}_{\mathbb{L}_{\mathrm{B}}}(\tau)$. Let G' and w' be the graph and its Hamiltonian cycle that are obtained by reducing statement "$\tau \in \mathbb{L}_{\mathrm{B}}$" to Hamiltonicity problem through the $\mathcal{NP}$ reduction.

Simulating messages Stage 2. If $\mathcal{S}$ obtained w during Stage 1, it interacts with V^* in the same way as an honest prover. Otherwise, $\mathcal{S}$ interacts with V^* as follows. The idea is that, since $\mathcal{S}$ know a witness w' for $G' \in \mathbf{L}_{\mathrm{HC}}$, $\mathcal{S}$ can correctly respond to the challenge for both $ch = 0$ and $ch = 1$ by committing to a random permutation of G' in the first step.

1. $\mathcal{S}$ chooses a random permutation π and computes $H := \pi(G')$. Then, $\mathcal{S}$ commits to the adjacency matrix $A = \{a_{i,j}\}_{i,j\in[q']}$ of H by using $\mathsf{GJS\text{-}Com}_{G,G',r}$. That is, $\mathcal{S}$ sends $c_{i,j} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(a_{i,j})$ to V^* for every $i, j \in [q']$.

   Let $\{\rho_{i,j}\}_{i,j\in[q']}$ be the randomness used in the GJS-Com commitments and $\pi(w')$ be the Hamiltonian cycle in H that is obtained by applying π on Hamiltonian cycle w' in G'.

2. $\mathcal{S}$ receives a random bit $ch \in \{0, 1\}$ from V^*.

3. **When $ch = 0$,** $\mathcal{S}$ sends π to V and decommits $c_{i,j}$ to $a_{i,j}$ honestly for every $i, j \in [q']$. That is, $\mathcal{S}$ sends $d_{i,j} := \mathsf{GJS\text{-}Dec}_{G,G',r}(c_{i,j}, a_{i,j}, \rho_{i,j})$ to V for every $i, j \in [q']$.

   When $ch = 1$, $\mathcal{S}$ decommits $c_{i,j}$ to 1 honestly for every i, j such that edge (i, j) is on the Hamiltonian cycle $\pi(w')$ in H. That is, for every such i and j, $\mathcal{S}$ sends $d_{i,j} := \mathsf{GJS\text{-}Dec}_{G,G',r}(c_{i,j}, a_{i,j}, \rho_{i,j})$ to V^*.

Simulating leakage queries in Stage 2. When V^* makes a leakage query f, $\mathcal{S}$ simulates the leakage as follows. Recall that in Stage 2-1, an honest prover commits to a $q' \times q'$ zero matrix whereas $\mathcal{S}$ commits to the adjacency matrix of H. Hence, $\mathcal{S}$ simulates the leakage by "explaining" the commitments to $\{a_{i,j}\}_{i,j\in[q']}$ as commitments to $\{0\}$ by using the adaptive security of GJS-Com and the knowledge of w'. Concretely, $\mathcal{S}$ does the following.

- First, for each $i, j \in [q']$, $\mathcal{S}$ constructs a function $F_{i,j}(\cdot)$ such that on input w, it outputs $\widetilde{\rho}_{i,j}$ such that $\mathsf{GJS\text{-}Com}_{G,G',r}(0; \widetilde{\rho}_{i,j}) = c_{i,j}$. Concretely, when $a_{i,j} = 0$, $F_{i,j}(\cdot)$ is a function that always outputs $\rho_{i,j}$, and when $a_{i,j} = 1$, $F_{i,j}(\cdot) \stackrel{\text{def}}{=} \mathsf{GJS\text{-}ExplainAsZero}_{G,G',r}(c_{i,j}, \cdot, w', \rho_{i,j})$.

- Next, $\mathcal{S}$ constructs a function $\widetilde{f}$ such that on input w, it computes $\mathsf{tape} := \mathsf{msgs} \| \{F_{i,j}(w)\}_{i,j\in[q']}$ and outputs $f(w, \mathsf{tape})$.

- Finally, $\mathcal{S}$ queries $\widetilde{f}$ to $\mathcal{L}_w$ and forwards the reply from $\mathcal{L}_w$ to V^*.

4.5.2.2 Amount of total leakage.

From the construction of $\mathcal{S}$, it always obtains at most the same amount of leakages as V^*. Hence, we have

$$\Pr\left[\mathsf{IDEAL}_{\mathcal{S}}(x, w_x, z) = \perp\right] = 0 \ .$$

4.5.2.3 Indistinguishability of views.

We show that for any cheating verifier V^* and any sequence $\{w_G\}_{G\in\mathbf{L}_{\mathrm{HC}}}$ such that $w_G \in \mathbf{R}_{\mathrm{HC}}(G)$, the following indistinguishability holds.

$$\{\mathsf{REAL}_{V^*}(G, w_G, z)\}_{G\in\mathbf{L}_{\mathrm{HC}}, z\in\{0,1\}^*} \stackrel{c}{\approx} \{\mathsf{IDEAL}_{\mathcal{S}}(G, w_G, z)\}_{G\in\mathbf{L}_{\mathrm{HC}}, z\in\{0,1\}^*} \ . \tag{4.5}$$

Toward this end, we consider the following hybrid experiments.

Hybrid $\mathbf{HYB}_0(G, z)$ is identical with experiment $\mathrm{REAL}_{V^*}(G, w, z)$. That is, V^* interacts with honest $P(G, w)$ and obtains leakage that is computed honestly based on witness w and the prover's randomness. The outputs of this hybrid is the view of V^*.

Hybrid $\mathbf{HYB}_1(G, z)$ is the same as HYB_0 except for the following.

- In Stage 1, an honest prover is replaced with the simulator. That is, c is computed by committing to $h(V^*)$, $\widehat{\beta}$ is computed by committing to β, and $\widehat{\delta}$ is computed by committing to δ.

  Let τ and $\bar{w}$ be the statement and the witness generated in it. Let G' and w' be the graph and its Hamiltonian cycle that are obtained by reducing statement "$\tau \in \mathbb{L}_{\mathrm{B}}$" to Hamiltonicity problem through the $\mathcal{NP}$ reduction.
- The leakage queries are answered by considering that the randomness generated by the prover during Stage 1 is equal to the messages sent to V^* during Stage 1.

Hybrid $\mathbf{HYB}_2(G, z)$ is the same as HYB_1 except for the following.

- As in $\mathcal{S}$, a random permutation π is chosen randomly at the beginning of Stage 2-1. Let $H \stackrel{\mathrm{def}}{=} \pi(G')$, and $A = \{a_{i,j}\}_{i,j\in[q']}$ be the adjacency matrix of H. Let $\pi(w')$ be the Hamiltonian cycle in H that is obtained by applying π on Hamiltonian cycle w' in G'.

  We remark that in this hybrid, the prover still commits to a $q'\times q'$ zero matrix as in HYB_1. Also, the leakage query immediately after Stage 2-1 is answered in exactly the same way as in HYB_1. In particular, when the leakage query is answered, π is not included in the randomness generated by the prover in Stage 2-1.
- In Stage 2-3, graph H_0 or H_1 is chosen as follows.

When $ch = 0$, $H_0 := H$.

When $ch = 1$, H_1 is the graph that is obtained by removing every edge in H except for the ones on Hamiltonian cycle $\pi(w')$.

The leakage query immediately after Stage 2-3 is answered in the same way as in HYB_1 by considering that H_0 or H_1 was chosen during Stage 2-3 as in HYB_1.

Hybrid $\mathbf{HYB}_3(G, z)$ is the same as HYB_2 except for the following.

- In Stage 2-1, for every $i, j \in [q']$, commitment $c_{i,j}$ is computed by committing to $a_{i,j}$ (instead of 0), i.e., $c_{i,j} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(a_{i,j})$.
- In Stage 2-3, for every $i, j \in [q']$, if commitment $c_{i,j}$ need to be decommitted, it is decommitted to $a_{i,j}$ honestly.
- When the leakage queries are answered during Stage 2, the randomness $\rho_{i,j}$ used for computing $c_{i,j}$ is simulated by $\widetilde{\rho}_{i,j}$ that is computed by function $F_{i,j}$ as in $\mathcal{S}$ for every $i, j \in [q']$.

Hybrid $\mathbf{HYB}_4(G, z)$ is identical with $\mathrm{IDEAL}_{\mathcal{S}}(x, w, z)$. That is, $\mathcal{S}(G, z)$ is executed given access to $\mathcal{L}_w$. The outputs of this hybrid is that of $\mathcal{S}$.

From a hybrid argument, we can obtain Equation (4.5) by showing that the outputs of each neighboring hybrids are indistinguishable. Let $\mathsf{HYB}_i(x, z)$ be the random variable representing the output of $\mathrm{HYB}_i(x, z)$ for each $i \in \{0, \ldots, 4\}$.

Claim 4.4. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_0(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} \overset{c}{\approx} \{\mathsf{HYB}_1(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} .$$

Proof. HYB_1 differs from HYB_0 only in that fake commitments of Com and $\mathsf{SWExtCom}$ are replaced with real commitments. Hence, the indistinguishability follows from the security of $\mathsf{Com}_{\mathrm{pub}}$ and C_{pub} (see Sections 4.3.1 and 4.3.6). □

Claim 4.5. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_1(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} \equiv \{\mathsf{HYB}_2(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} .$$

Proof. This claim can be proven by inspection. Observe that HYB_2 differs from HYB_1 only in the way graph H_0 or H_1 is chosen in Stage 2. When $ch = 0$, the distribution of H_0 in HYB_2 is the same as that in HYB_1 since H_0 is obtained both in HYB_2 and HYB_1 by applying a random permutation on G'. When $ch = 1$, the distribution of H_1 in HYB_2 is the same as that in HYB_1 since the Hamiltonian cycle w' in G' is mapped to a random q-cycle by π. Hence, the output of HYB_2 is identically distributed with that of HYB_1. □

Claim 4.6. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_2(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} \overset{c}{\approx} \{\mathsf{HYB}_3(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} .$$

Proof. Assume for contradiction that for infinitely many $G \in \mathbf{L}_{\mathrm{HC}}$, there exists $z \in \{0,1\}^*$ such that a distinguisher $\mathcal{D}$ distinguishes $\mathsf{HYB}_2(G,z)$ and $\mathsf{HYB}_3(G,z)$ with advantage $1/p(n)$ for a polynomial $p(\cdot)$. Fix any such G and z. To derive a contradiction, we consider the following intermediate hybrids.

Hybrid $\mathbf{HYB}_{2:0}(G,z)$ is identical with $\mathrm{HYB}_2(G,z)$.

Hybrid $\mathbf{HYB}_{2:k}(G,z)$, where $k \in [q'^2]$, is the same as $\mathrm{HYB}_{2:k-1}$ except for the following. Let $u, v \in [q']$ be such that $(u-1)q' + v = k$.

- In Stage 2-1, commitment $c_{u,v}$ is computed by committing to $a_{u,v}$ (instead of 0), i.e., $c_{u,v} \leftarrow \mathsf{GJS\text{-}Com}_{G,G',r}(a_{u,v})$.
- In Stage 2-3, if commitment $c_{u,v}$ need to be decommitted, it is decommitted to $a_{u,v}$ honestly.
- When the leakage queries are answered during Stage 2, the randomness $\rho_{u,v}$ used for computing $c_{u,v}$ is simulated by $\widetilde{\rho}_{u,v}$ that is computed by function $F_{u,v}$ as in $\mathcal{S}$.

Clearly, $\mathrm{HYB}_{2:q'^2}$ is identical with HYB_3. Hence, there exists $k^* \in [q'^2]$ such that the output of $\mathrm{HYB}_{2:k^*-1}$ and that of $\mathrm{HYB}_{2:k^*}$ can be distinguished with advantage $1/q'^2 p(n)$. Furthermore, from an average argument, there exists a prefix σ of the execution of HYB_{k^*-1} up until permutation π is chosen in Stage 2-1 (i.e., just before $\{c_{i,j}\}_{i,j\in[q']}$ is sent to V^*) such that under the condition that a prefix of the execution is σ, the output of $\mathrm{HYB}_{2:k^*-1}$ and that of $\mathrm{HYB}_{2:k^*}$ can be distinguished with advantage $1/q'^2 p(n)$. Notice that σ determines G', w', r, $\{a_{i,j}\}_{i,j\in[q']}$.

We derive a contradiction by showing that we can break the adaptive security of $\mathsf{GJS\text{-}Com}$ (Lemma 4.8). Specifically, we show that $\mathsf{Exp}_0^{\mathrm{GJS}}(n,G,G',w,w',r)$ and $\mathsf{Exp}_1^{\mathrm{GJS}}(n,G,G',w,w',r)$ can be distinguished with advantage $1/q'^2 p(n)$. Toward this end, consider the following distinguisher $\mathcal{D}'$.

- Externally, $\mathcal{D}'$ takes (c,ρ_0,d_1) as well as (n,G,G',w,w',r) as input. $\mathcal{D}'$ also takes (σ,z) as non-uniform input.
- Internally, $\mathcal{D}'$ invokes V^* and simulates $\mathrm{HYB}_{2:k^*-1}(G,z)$ for V^* from σ honestly except for the following. Let $u^*, v^* \in [q']$ be such that $(u^*-1)q' + v^* = k^*$. Notice that it must hold that $a_{u^*,v^*} = 1$ since $\mathrm{HYB}_{2:k^*}$ is identical with $\mathrm{HYB}_{2:k^*-1}$ when $a_{u^*,v^*} = 0$.
  - In Stage 2-1, commitment c_{u^*,v^*} is defined by setting $c_{u^*,v^*} := c$.
  - In Stage 2-3, when commitment c_{u^*,v^*} is decommitted, it is decommitted to $a_{u^*,v^*} = 1$ by sending d_1.
  - When the leakage queries are answered during Stage 2, the randomness ρ_{u^*,v^*} used for computing c_{u^*,v^*} is simulated by setting $\widetilde{\rho}_{u^*,v^*} := \rho_0$.

  Let view be the view of V^*. Then, $\mathcal{D}'$ outputs $\mathcal{D}(\mathsf{view})$.

When $(c, \rho_0, d_1) \leftarrow \mathsf{Exp}_0^{\mathrm{GJS}}(n, G, G', w, w', r)$ (i.e., when c is a commitment to 0, ρ_0 is the randomness that is used to generate c, and d_1 is a decommitment to 1 that is computed by GJS-EquivToOne), $\mathcal{D}'$ emulates $\mathrm{HYB}_{2:k^*-1}$ for V^* perfectly. On the other hand, when $(c, \rho_0, d_1) \leftarrow \mathsf{Exp}_1^{\mathrm{GJS}}(n, G, G', w, w', r)$ (i.e., when c is a commitment to 1, ρ_0 is randomness that is computed by GJS-ExplainAsZero, and d_1 is a decommitment to 1 that is computed honestly), $\mathcal{D}'$ emulates $\mathrm{HYB}_{2:k^*}$ for V^* perfectly. Hence, from our assumption, $\mathcal{D}'$ distinguishes $\mathsf{Exp}_0^{\mathrm{GJS}}(n, G, G', w, w', r)$ and $\mathsf{Exp}_1^{\mathrm{GJS}}(n, G, G', w, w', r)$ with advantage $1/q'^2 p(n)$, and therefore we reach a contradiction. □

Claim 4.7. *We have the following indistinguishability.*

$$\{\mathsf{HYB}_3(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} \equiv \{\mathsf{HYB}_4(G, z)\}_{G \in \boldsymbol{L}_{\mathrm{HC}}, z \in \{0,1\}^*} \quad .$$

Proof. In HYB_3, the prover interacts with V^* in exactly the same way as $\mathcal{S}$. Hence, the claim follows. □

From Claim 4.4, 4.5, 4.6, and 4.7, we obtain Equation (4.5). This concludes the proof of Lemma 4.10.

□

This concludes the proof of Theorem 4.1. □

Chapter 5

Non-black-box Zero-knowledge in the Fully Concurrent Setting

In this chapter, we show our third result: A new construction of non-black-box concurrent zero-knowledge arguments.

5.1 Background

As one can see in Definition 2.11, the zero-knowledgeness of interactive proofs/arguments is defined by using the *simulation paradigm*: An interactive proof/argument is defined to be zero-knowledge if for any adversarial verifier there exists a *simulator* that can output a simulated view of the adversary.

Traditionally, the security of ZK protocols was proven via *black-box simulation*. That is, the zero-knowledge property was proven by showing a simulator that uses the adversary only as an oracle. Since black-box simulators use the adversaries only as oracles, their advantage is very limited—essentially, their only advantage is the ability to rewind the adversaries. Nonetheless, black-box simulation is actually quite powerful, and it has been used to obtain ZK protocols with a variety of additional properties, security, and efficiency.

However, black-box simulation has inherent limitations. For example, let us consider *public-coin* ZK *protocols* and *concurrent* ZK *protocols*, where the former is the ZK protocols such that the verifier sends only the outcome of its coin-tossing during the protocols, and the latter is the ZK protocols such that their zero-knowledge property holds even when they are concurrently executed many times. It is known that both of them can be constructed by using black-box simulation techniques [GMW91, RK99, KP01, PRS02]. However, it is also known that neither of them can be constructed by black-box simulation techniques if we additionally require round efficiency. Specifically, it was shown that constant-round public-coin ZK protocols and $o(\log n/\log\log n)$-round concurrent ZK protocols cannot be proven secure via black-box simulation [GK96b, CKPR02]. Furthermore, it was also shown that no public-coin concurrent ZK protocol can be proven secure via black-box simulation irrespective to its round complexity [PTW09].

Natural questions to ask are whether the ZK property can be proven by using *non-black-box simulation techniques*, and whether the limitations of black-box simulation can be overcome by using non-black-box simulation. Non-black-box simulation techniques are, however, significantly hard to develop. Specifically, non-black-box simulation seems to inherently involve "reverse engineering" of the adversaries, and such reverse engineering seems very difficult.

Barak [Bar01] made a breakthrough about non-black-box stimulation by proposing the first non-black-box simulation technique under a standard assumption, and showing that a black-box impossibility result can be overcome by using it. Specifically, Barak used his non-black-box simulation technique to obtain a constant-round public-coin ZK protocol under the assumption that a family of collision-resistant hash functions exists. (Recall that, as noted above, constant-round public-coin ZK protocols cannot be proven secure via black-box simulation.) The simulation technique of Barak is completely different from previous ones. Specifically, in his simulation technique, the simulator runs in a "straight-line" manner (that is, it does not rewind the adversary) and simulates the adversary's view by using the code of the adversary.[31]

Non-black-box simulation in the concurrent setting. Since Barak's non-black-box simulation technique allows us to overcome a black-box impossibility result, it is natural to ask whether we can overcome other black-box impossibility results as well by using Barak's technique. In particular, since Barak's simulation technique works in a straight-line manner and therefore completely removes the issue of *recursive rewinding* [DNS04] that arises in the setting of concurrent ZK, it is natural to expect that Barak's simulation technique can be used to overcome the black-box impossibility results of $o(\log n/\log\log n)$-round concurrent ZK protocols and public-coin concurrent ZK protocols.

However, it turned out that Barak's non-black-box simulation technique is hard to use in the concurrent setting. In fact, although Barak's technique can be extended so that it can handle *bounded-concurrent execution* [Bar01] (i.e., concurrent execution where the protocol is concurrently executed a bounded number of times) and *parallel execution* [PRT13], it had been open for years to extend it so that it can handle fully concurrent execution. An important step towards obtaining non-black-box simulation in the fully concurrent setting was made by Deng, Goyal, and Sahai [DGS09], who used Barak's technique in the fully concurrent setting by combining it with a black-box simulation technique (specifically, with the recursive rewinding technique of Richardson and Kilian [RK99]). Another important step was made by Bitansky and Paneth [BP12, BP13, BP15], who developed a new non-black-box simulation technique (which is *not* based on that of Barak) that can handle fully concurrent execution when being combined with a black-box simulation technique (again, the recursive rewinding technique of [RK99]). The simulation techniques of these works are powerful enough to allow us to overcome another black-box impossibility result (namely the impossibility of *simultaneously resettable* ZK *protocols* [CGGM00, BGGL01]). However, they are not strict improvement over Barak's non-black-box simulation technique

[31] The notion of "straight-line simulation" is, unfortunately, hard to formalize. I this thesis, we use it only informally.

since they do not have some of the useful properties that Barak's technique do have, such as the public-coin property and the straight-line simulation property. As a result, they do not immediately allow us to overcome the black-box impossibility results of $o(\log n/\log\log n)$-round concurrent ZK protocols and public-coin concurrent ZK protocols.

Recently, several works showed that with a trusted setup or non-standard assumptions, Barak's simulation technique can be extended so that it can handle fully concurrent execution (without losing its public-coin property and straight-line simulation property). Furthermore, they showed that with their versions of Barak's technique, it is possible to overcome the black-box impossibility results of $o(\log n/\log\log n)$-round concurrent ZK protocols and public-coin concurrent ZK protocols. For example, Canetti et al. [CLP13a] constructed a public-coin concurrent ZK protocol in the *global hash function (GHF) model*, where a single hash function is used in all concurrent sessions. Also, Chung et al. [CLP13b] constructed a constant-round concurrent ZK protocol by assuming the existence of $\mathcal{P}$*-certificates* (i.e., "succinct" non-interactive proofs/arguments for $\mathcal{P}$), Pandey et al. [PPS15] constructed a constant-round concurrent ZK protocols by assuming the existence of *differing-input indistinguishability obfuscators*, and Chung et al. [CLP15] constructed a constant-round concurrent ZK protocols by assuming the existence of indistinguishability obfuscators.

Very recently, Goyal [Goy13] showed that Barak's non-black-box simulation technique can be extended so that it can handle fully concurrent execution *even in the plain model under standard assumptions*. Goyal then used his version of Barak's technique to obtain the first public-coin concurrent ZK protocol in the plain model under a standard assumption (the existence of a family of collision-resistant hash functions), where its round complexity is $O(n^{\epsilon})$ for an arbitrary constant $\epsilon > 0$. Like the original simulation technique of Barak (and many of its variants), the simulation technique of Goyal has a straight-line simulator; hence, Goyal's simulator performs *straight-line concurrent simulation*. Because of this straight-line concurrent simulation property, the simulation technique of Goyal has huge potential. In fact, it was shown subsequently that Goyal's technique can be used to obtain new results on concurrently secure multi-party computation and concurrent blind signatures [GGS15].

In summary, we currently have several positive results on non-black-box simulation in the concurrent setting, and in particular we have a one that has a straight-line concurrent simulator in the plain model under a standard assumption [Goy13]. However, the state-of-the-art is still not satisfactory and there are still many open problems to be addressed. For example, the simulation technique of Goyal [Goy13] requires the protocol to have $O(n^{\epsilon})$ rounds, so the problem of constructing $o(\log n/\log\log n)$-round concurrent ZK protocols in the plain model under standard assumptions is still open. Thus, studying more on non-black-box simulation and developing new non-black-box simulation techniques in the concurrent setting is still an important research direction.

5.1.1 Our Result

In this chapter, we propose a new variant of Barak's non-black-box simulation technique and use it to give a new proof of the following theorem, which was originally proven

by Goyal [Goy13].

Main Theorem. *Assume the existence of a family of collision resistant hash functions. Then, for any constant $\epsilon > 0$, there exists an $O(n^{\epsilon})$-round public-coin concurrent zero-knowledge argument of knowledge.*

Like the simulation technique of Goyal, our simulation technique can handle fully concurrent execution in the plain model under a standard assumption, and it has a simulator that runs in a straight-line manner in the fully concurrent setting. We emphasize that our simulation technique requires the same hardness assumption and the same round complexity as that of Goyal; hence, it does not immediately lead to improvement over the result of Goyal. Nevertheless, we believe that our simulation technique is interesting because it is different from that of Goyal and its analysis is (in our opinion) simpler than the analysis of Goyal's technique. (A comparison between our simulation technique and that of Goyal is given in Section 5.2.3.) We hope that our technique leads to further study on non-black-box simulation in the concurrent setting.

Brief overview of our technique. Our public-coin concurrent ZK protocol is based on the public-coin concurrent ZK protocol of Canetti, Lin, and Paneth (CLP) [CLP13a], which is secure in the global hash function model. Below, we give a brief overview of our technique under the assumptions that the readers are familiar with Barak's non-black-box simulation technique and CLP's techniques. In Section 5.2, we give a more detailed overview of our technique, including the explanation of the techniques of Barak and CLP.

The protocol of CLP is similar to the ZK protocol of Barak except that it has multiple "slots" (i.e., pairs of a prover's commitment and a receiver's random-string message). A key observation by CLP is that given multiple slots, one can avoid the blow-up of the simulator's running time, which is the main obstacle to use Barak's simulation technique in the concurrent setting. More precisely, CLP's observation is that given multiple slots, the simulator can use any of these slots when generating the PCP proof in the universal argument (UA) of Barak's protocol, and therefore it can avoid the blow-up of its running time by using a good "proving strategy" that determines which slots to use in the generation of the PCP proofs in concurrent sessions. The proving strategy that CLP use is similar in spirit to the *oblivious rewinding strategy* [KP01, PRS02] of black-box concurrent ZK protocols. In particular, in the proving strategy of CLP, the simulator recursively divides the simulated transcript between honest provers and the cheating verifier into "blocks," and generates the PCP proofs only at the end of the blocks.

A problem that CLP encountered is that the simulator has only one opportunity to give the UA proof in each session, and thus it need to remember all previously generated PCP proofs if the adversary delays the execution of the UA proofs in all sessions. Because of this problem, the length of the PCP proofs can be rapidly blowing up in the concurrent setting, and the size of the simulator cannot be bounded by a polynomial. In [CLP13a], CLP solved this problem in the global hash function model by cleverly using the global hash function in UA.

To solve this problem in the plain model, we modify the protocol of CLP so that the simulator has multiple opportunities to give the UA proof in each session. We then show that by using a good proving strategy that also determines which opportunity the simulator takes to give the UA proof in each session, the simulator can avoid the blow-up of its size as well as that of its running time. Our proving strategy guarantees that a PCP proof generated at the end of a block is used only in its "parent block"; because of this guarantee, the simulator need to remember each PCP proof only for a limited time and therefore the length of the PCP proofs does not blow up. This proving strategy is the core of our simulation technique and the main deference between the simulation technique of ours and that of Goyal [Goy13]. (The simulator of Goyal also has multiple opportunities to give the UA proof in each session, but it determines which opportunity to take by using a proving strategy that is different from ours.) Interestingly, the strategy that we use is deterministic (whereas the strategy that Goyal uses is probabilistic). Because of the use of this deterministic strategy, we can analyze our simulator in a relatively simple way. In particular, when showing that every session is always successfully simulated, we need to use only a simple counting argument.

5.1.2 Outline

In Section 5.2, we give an overview of our techniques. In Section 5.3, we give the notations and definitions that are used specifically in this chapter. In Section 5.4, we describe our non-black-box concurrent ZK argument and prove its security.

5.2 Overview of Our Techniques

As mentioned in Section 5.1.1, our protocol is based on the protocol of Canetti et al. [CLP13a], which in turn is based on Barak's non-black-box zero-knowledge protocol [Bar01]. Below, we first recall the protocols of [Bar01, CLP13a] and then give an overview of our protocol.

5.2.1 Known Techniques

Barak's protocol. Roughly speaking, Barak's non-black-box zero-knowledge argument BarakZK proceeds as follows.

Protocol BarakZK

1. The verifier V chooses a hash function $h \in \mathcal{H}_n$ and sends it to the prover P.
2. P sends $c \leftarrow \mathsf{Com}(0^n)$ to V, where Com is a statistically binding commitment scheme. (For simplicity, in this overview we assume that Com is non-interactive.) Then, V sends a random string $r \in \{0,1\}^n$ to P. In the following, the pair (c, r) is called a *slot*.
3. P proves the following statement by using a witness-indistinguishable argument.
   - $x \in L$, or

- $(h, c, r) \in \Lambda$, where Λ is a language such that $(h, c, r) \in \Lambda$ holds if and only if there exists a machine Π such that (i) c is a commitment to $h(\Pi)$ and (ii) Π outputs r within $n^{\log \log n}$ steps.[32]

Since polynomial-time algorithms cannot check whether or not Π outputs r within $n^{\log \log n}$ steps, the statement proven in Step 3 is not in $\mathcal{NP}$. Thus, P proves this statement by using a witness-indistinguishable *universal argument* (WIUA), which is, roughly speaking, a witness-indistinguishable argument for $\mathcal{NEXP}$ such that the prover's running time is bounded by poly(Time(w)) when the prover uses a witness w during the proof, where Time(w) is the time that is needed for verifying the validity of w.

Roughly speaking, the security of BarakZK is proven as follows. The soundness is proven by observing that even when a cheating prover P^* commits to $h(\Pi)$ for a machine Π, we have $\Pi(c) \neq r$ with overwhelming probability because r is chosen after P^* commits to $h(\Pi)$. The zero-knowledge property is proven by using a simulator that commits to a machine Π that emulates the cheating verifier V^*; since $\Pi(c) = V^*(c) = r$ from the definition, the simulator can give a valid proof in WIUA. Such a simulator runs in polynomial time since, from the property of WIUA, the running time of the simulator during WIUA is bounded by poly(t), where t is the running time of $\Pi(c)$.

Barak's protocol in the concurrent setting. A limitation of BarakZK is that we do not know how to prove its zero-knowledge property in the concurrent setting. Recall that in the concurrent setting, a protocol is executed many times concurrently; hence, to prove the zero-knowledge property of a protocol in the concurrent setting, we need to design a simulator against cheating verifiers that participate in many *sessions* of the protocol with honest provers. The above simulator for BarakZK, however, does not work against such verifiers since $V^*(c) = r$ does not hold when a verifier V^* participates in other sessions during a slot of a session (i.e., $V^*(c) \neq r$ holds when V^* first receives c in a session, next receives messages in other sessions, and then sends r in the first session).

A potential approach to proving the concurrent zero-knowledge property of BarakZK is to use a simulator $\mathcal{S}$ that commits to a machine that emulates $\mathcal{S}$ itself. The key observation behind this approach is the following: When V^* participates in other sessions during a slot of a session, all the messages that V^* receives in the other sessions are actually generated by $\mathcal{S}$; hence, if the committed machine Π can emulate $\mathcal{S}$, it can emulate all the messages between c and r for V^*, so $\Pi(c)$ can output r even when V^* receives many messages during a slot.[33]

This approach however causes a problem in the simulator's running time. For example, let us consider the following "nested concurrent sessions" schedule (see Figure 5.1).

- The $(i + 1)$-th session is executed in such a way that it is completely contained in

[32] Here, $n^{\log \log n}$ can be replaced with any super-polynomial function. We use $n^{\log \log n}$ for concreteness.

[33] The circularity about the simulator committing to a machine that emulates the simulator itself can be avoided by separating it to the main simulator $\mathcal{S}$ and an auxiliary simulator aux-$\mathcal{S}$. Roughly speaking, aux-$\mathcal{S}$ takes a code of a machine Π as input and does simulation by committing to Π; then, $\mathcal{S}$ invokes aux-$\mathcal{S}$ with input Π = aux-$\mathcal{S}$.

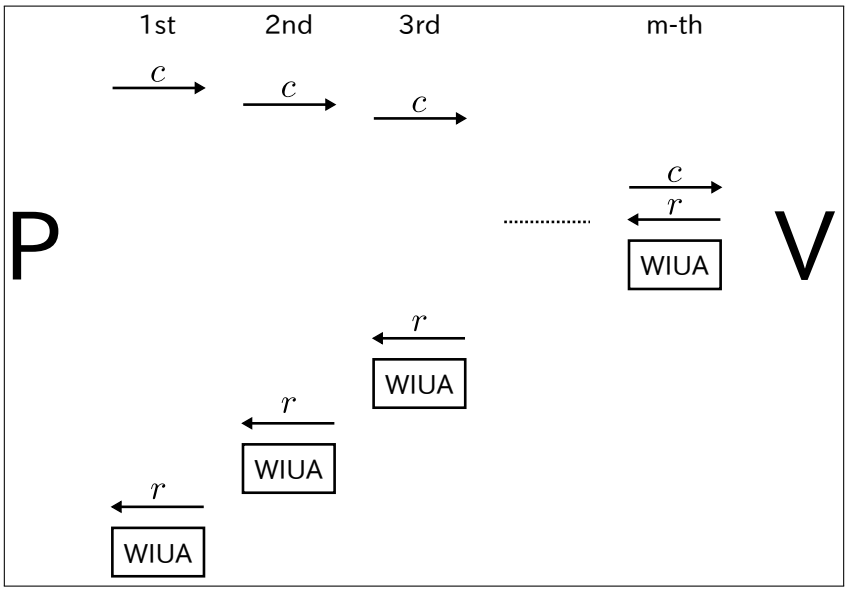

Figure 5.1: The "nested concurrent sessions" schedule.

the slot of the i-th session. That is, V^* starts the $(i+1)$-th session after receiving c in the i-th session, and sends r in the i-th session after completing the $(i+1)$-th session.

Let m be the number of sessions, and let t be the running time of $\mathcal{S}$ during the simulation of the m-th session. Then, to simulate the $(m-1)$-th session, $\mathcal{S}$ need to run at least $2t$ steps—t steps for simulating the slot (which contains the m-th session) and t steps for simulating WIUA. Then, to simulate the $(m-2)$-th session, $\mathcal{S}$ need to run at least $4t$ steps—$2t$ steps for simulating the slot and $2t$ steps for simulating WIUA. In general, to simulate the i-th session, $\mathcal{S}$ need to run at least $2^{m-i}t$ steps. Thus, the running time of $\mathcal{S}$ becomes super-polynomial when $m = \omega(\log n)$.

Protocol of Canetti et al. [CLP13a]. To avoid the blow-up of the simulator's running time, Canetti, Lin, and Paneth (CLP) [CLP13a] used the "multiple slots" approach, which was originally used in black-box concurrent zero-knowledge protocols [RK99, KP01, PRS02]. The idea is that if BarakZK has multiple sequential slots, $\mathcal{S}$ can choose any of them as a witness in WIUA, and therefore $\mathcal{S}$ can avoid the nested computations in WIUA by using a good *proving strategy* that determines which slot to use as a witness in each session. To implement this approach, CLP first observed that the four-round public-coin UA of Barak and Goldreich [BG08], from which WIUA can be constructed, can be divided into the *offline phase* and the *online phase* such that all heavy computations are done in the offline phase. Concretely, CLP divided the UA of [BG08] as follows. Let $x \in L$ be the statement to be proven in UA and w be a witness for $x \in L$.

Offline/online UA

- Offline Phase:

1. V sends a random hash function $h \in \mathcal{H}_n$ to P.
2. P generates a PCP proof π of statement $x \in L$ by using w as a witness, and then computes $\mathsf{UA}_2 := h(\pi)$. In the following, (h, π, UA_2) is called the *offline proof*.

- Online Phase:

1. P sends UA_2 to V.
2. V chooses randomness ρ for the PCP-verifier and sends $\mathsf{UA}_3 := \rho$ to P.
3. P computes a PCP-query Q by executing the PCP-verifier with statement $x \in L$ and randomness ρ, and then sends $\{\pi_i\}_{i \in Q}$ to V (i.e., partially reveals π according to the locations that are specified by Q) while proving that $\{\pi_i\}_{i \in Q}$ is correctly computed w.r.t. the string it hashed in UA_2. (Such a proof can be generated efficiently if P computes $\mathsf{UA}_2 = h(\pi)$ by tree hashing.)
4. V first verifies the correctness of the revealed bits $\{\pi_i\}_{i \in Q}$, and next verifies the PCP proof by executing the PCP-verifier on $\{\pi_i\}_{i \in Q}$.

Note that the only heavy computations—the generation of π and the computation of $h(\pi)$—are performed in the offline phase; the other computations can be performed in a fixed polynomial time. (For simplicity, here we assume that P has random access to π.[34]) Thus, in the online phase, the running time of P can be bounded by a fixed polynomial in n. In the offline phase, the running time of P is bounded by a fixed polynomial in t, where t is the time needed for verifying $x \in L$ with witness w. The length of the offline proof is also bounded by a polynomial in t.

CLP then considered the following protocol (which is an over-simplified version of their final protocol). Let N_{slot} be a parameter that is determined later.

Protocol BasicCLP

Stage 1. V chooses a hash function $h \in \mathcal{H}_n$ and sends it to P.

Stage 2. For each $i \in [N_{\mathrm{slot}}]$ in sequence, P and V do the following.

- P sends $C_i \leftarrow \mathsf{Com}(0^n)$ to V. Then, V sends a random string $r_i \in \{0,1\}^n$ to P.

Stage 3. P and V execute the special-purpose WIUA of Pass and Rosen [PR05b] with the UA system of Barak and Goldreich [BG08] being used as the underlying UA system. Concretely, P and V do the following.

1. P sends $D_{\mathsf{UA}} \leftarrow \mathsf{Com}(0^n)$ to V.
2. V sends a third-round UA message UA_3 to P (i.e., V sends a random string of appropriate length).
3. P proves the following statement by using a witness-indistinguishable proof of knowledge (WIPOK).

[34] This assumption is used only in this overview.

- $x \in L$, or
- there exist $i \in [N_{\mathsf{slot}}]$ and a second- and a fourth-round UA message $\mathsf{UA}_2, \mathsf{UA}_4$ such that D_{UA} is a commitment to UA_2 and $(h, \mathsf{UA}_2, \mathsf{UA}_3, \mathsf{UA}_4)$ is an accepting proof for the statement $(h, C_i, r_i) \in \Lambda$.

Recall that the idea of the multiple-slot approach is that $\mathcal{S}$ avoids nested computations in WIUA by using a good proving strategy that determines which slots to use as witnesses. Based on this idea, CLP designed a proving strategy and a simulator as follows. First, their simulator works roughly as follows: $\mathcal{S}$ commits to a machine in each slots, where the committed machines emulate $\mathcal{S}$ as mentioned above; $\mathcal{S}$ then computes an offline proof (including a PCP proof) w.r.t. a slot that is chosen according to a proving strategy; $\mathcal{S}$ then commits to the second-round UA message (i.e., the hash vlaue of the PCP proof) in Stage 3-1 and gives a WIPOK proof in Stage 3-3 using the offline proof as a witness. Second, their proving strategy works roughly as follows. As in the *oblivious rewinding strategy* of black-box concurrent zero-knowledge protocols [KP01, PRS02], the proving strategy of CLP recursively divides the entire transcript between honest provers and the cheating verifier into "blocks." Let M be the total number of messages and q be a parameter called the *splitting factor*. Assume for simplicity that M is a power of q, i.e., $M = q^d$ for $d \in \mathbb{N}$.

- The level-d block is the entire transcript. Thus, the level-d block contains $M = q^d$ messages.
- Then, the level-d block is divided into q sequential blocks, where each of them contains q^{d-1} messages. These blocks are called the level-$(d-1)$ blocks.
- Similarly, each level-$(d-1)$ block is divided into q sequential blocks, where each of them contains q^{d-2} messages. These blocks are called the level-$(d-2)$ blocks.
- In this way, each block is continued to be divided into q blocks until the level-0 blocks are obtained. A level-0 block contains only a single message.

Then, the proving strategy of CLP specifies that at the end of each block in each level, $\mathcal{S}$ computes offline proofs w.r.t. each slot that is contained in that block. Note that the offline proofs are computed only at the end of the blocks, and the maximum level of the blocks (i.e., d) is constant when $q = n^\epsilon$ for a constant ϵ. We therefore have at most constant levels of nesting in the executions of WIUA. Furthermore, it was shown by CLP that when $N_{\mathsf{slot}} = \omega(q) = \omega(n^\epsilon)$, the simulator does not "get stuck," i.e., in every session, the simulator obtains an offline proof before Stage 3 starts.

The protocol $\mathsf{BasicCLP}$ is, however, not concurrent zero-knowledge in the plain model. Roughly speaking, this is because the size of $\mathcal{S}$'s state can become super-polynomial. Recall that $\mathcal{S}$ generates an offline proof in Stage 2 and uses it in Stage 3 in each session. Then, since V^* can choose any concurrent schedule (and in particular can delay the execution of Stage 3 arbitrarily), in general, $\mathcal{S}$ need to remember every previously generated offline proof during its execution. This means that each committed machine also need to contain every previously generated offline proof (otherwise they cannot emulate the simulator), and therefore an offline proof (which is generated

by using a committed machine as a witness) is as long as the total length of all the offline proofs that are generated previously. Thus, the length of the offline proofs can be rapidly blowing up and the size of $\mathcal{S}$'s state cannot be bounded by a polynomial.

A key observation by CLP is that this problem can be solved in the *global hash model*, in which a global hash function is shared by all sessions. Roughly speaking, CLP avoided the blow-up of the simulator's size by considering machines that contain only the hash values of the offline proofs; then, to guarantee that the simulation works with such machines, they modified BasicCLP in such a way that P proves in WIUA that $x \in L$ or the committed machine outputs r given access to the *hash-inversion oracle*; in the simulation, $\mathcal{S}$ commits to a machine that emulates $\mathcal{S}$ by recovering offline proofs from their hash values using the hash-inversion oracle. In this modified protocol, the soundness is proven by using the fact that the same hash function is used across all the sessions.

In this way, CLP obtained a public-coin concurrent zero-knowledge protocol in the global hash model. Since $q = n^{\epsilon}$ and $N_{\mathrm{slot}} = \omega(q)$, the round complexity is $O(n^{\epsilon'})$ for a constant ϵ'. (Since ϵ is an arbitrary constant, ϵ' can be an arbitrary small constant.) CLP also showed that by modifying the protocol further, the round complexity can be reduced to $O(\log^{1+\epsilon} n)$.

5.2.2 Our Techniques

We obtain our $O(n^{\epsilon})$-round protocol by modifying BasicCLP of Canetti et al. [CLP13a] so that its concurrent zero-knowledge property can be proven without using global hash functions. Recall that a global hash function is used in [CLP13a] to avoid the blow-up of the simulator's state size. In particular, a global hash function is used so that the simulation works even when the committed machines do not contain any previously computed offline proof. Below, we first introduce the machines that our simulator commits to in the slots. They do not contain any previously generated offline proof and therefore their sizes are bounded by a fixed polynomial. We then explain our protocol and simulator, which are designed so that the simulation works even when the committed machines do not contain any previously computed offline proof. In the following, we set $q := O(n^{\epsilon})$, $N_{\mathrm{slot}} := \omega(q)$, and $N_{\mathrm{col}} := \omega(1)$.

The machines to be committed. Our first observation is that if the committed machines emulate a larger part of the simulation, they generate more offline proofs by itself and therefore are more likely to be able to output r even when they contain no offline proof. For example, let us consider an extreme case that the committed machines emulate the simulator from the beginning of the simulation (rather than from the beginning of the slots in which they are committed to). In this case, the committed machines generate every offline proof by themselves, so they can output r even when they contain no offline proof. A problem of this case is that the running time of each committed machine is too long and the running time of the simulator becomes super-polynomial.

Based on this observation, we consider machines that emulate the simulator from the beginning of the "current blocks," i.e., machines that emulate the simulator from the beginning of the blacks that contain the commitments in which they are committed to.

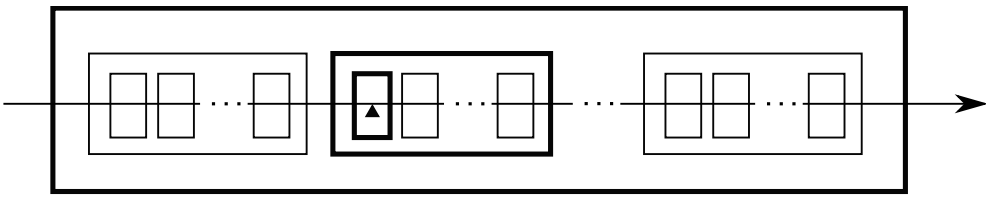

Figure 5.2: An illustration of the current blocks. When the next scheduled message is located on the place specified by the triangle, the current blocks are the ones described with the thick lines.

More precisely, we first modify BasicCLP so that P gives N_{col} parallel commitments in each slot. Then, our simulator commits to machines in each slot as follows. Below, the i-th *column* ($i \in [N_{\mathrm{col}}]$) of a slot is the i-th commitment of the slot, and the *current level-ℓ block* ($\ell \in [d]$) at a point during the interaction with V^* is the level-ℓ block that will contain the next-scheduled message (see Figure 5.2).

- In the i-th column ($i \in [d]$) of a slot, our simulator commits to a machine Π_i that emulates the simulator from the beginning of the current level-i block, where Π_i does not contain any offline proofs and it terminates with output fail if the emulation fails due to the lack of the offline proofs.

Now, we observe that the simulator's running time does not blow-up when the simulator commits to machines as above. Assume that, as in the proving strategy of CLP, the simulator computes the offline proofs only at the end of the blocks. Specifically, assume that the simulator compute the offline proofs at the end of the blocks as follows.

- At the end of a level-ℓ block b ($\ell \in [d]$), the simulator finds all the slots that are contained in block b, and generates offline proofs w.r.t. those slots by using the machine that are committed to in their ℓ-th columns. Note that those committed machines emulate the simulator from the beginning of block b, so the simulator can indeed use them as witness when generating the offline proofs.

Let t_i be the maximum time needed for simulating a level-i block ($i \in \{0, \ldots, d\}$). Recall that a level-i block consists of q level-$(i-1)$ blocks, and at most $m := \mathsf{poly}(n)$ offline proofs are generated at the end of each level-$(i-1)$ block. Then, since each offline proof at the end of a level-$(i-1)$ block can be computed in $\mathsf{poly}(t_{i-1})$ steps, we have

$$t_i \leq q \cdot (t_{i-1} + m \cdot \mathsf{poly}(t_{i-1})) \leq \mathsf{poly}(t_{i-1}) \ .$$

Recall that we have $t_0 = \mathsf{poly}(n)$ (this is because a level-0 block contains only a single message), and the maximum level $d = \log_q M$ is constant. We therefore have $t_d = \mathsf{poly}(n)$, so the running time of the simulator is bounded by a polynomial in n.

We note that although the above machines do not contain any previously generated offline proof, they do contain every previously generated WIPOK witness (i.e., UA_2 and UA_4).[35] As explained below, allowing the committed machines to contain every previously generated WIPOK witness is crucial to obtain our protocol and simulator.

[35] Since the length of the WIPOK witnesses is bounded by a fixed polynomial, the sizes of the committed machines do not blow up even when they contain every previously generated WIPOK witness.

Our protocol and simulator. When the simulator commits to the above machines, the simulation does not work if the committed machines output fail. In particular, the simulation fails if there exists a block in which the simulator uses an offline proof that are generated before the beginning of that block. (If such a block exists, the machines that are committed in this block output fail since they cannot emulate the simulator due to the lack of the offline proof.) Thus, to guarantee successful simulation, we need to make sure that in each block, the simulator uses only the offline proofs that are generated in that block. Of course, we also need to make sure that the simulator does not "get stuck," i.e., we need to guarantee that in each session, the simulator obtains a valid witness before WIPOK starts.

To avoid the simulation failure, we first modify BasicCLP as follows. As noted in the previous paragraph, we need to construct a simulator such that in each block, it uses only the offline proofs that are generated in that block. In BasicCLP, it is hard to construct such a simulator since offline proofs may be used long after they are generated. (Recall that during the simulation, the offline proofs are generated in Stage 2 and they are used in Stage 3 to compute WIPOK witnesses, and V^* can delay the execution of Stage 3 arbitrarily.) Thus, we modify BasicCLP so that the simulator can use the offline proofs soon after generating them; in particular, we modify BasicCLP so that Stage 3 can be executed "in the middle of" Stage 2. Concretely, after each slot in Stage 2, we add another slot, *a* UA-*slot*, that can be used for executing Stage 3-1 and Stage 3-2. That is, we consider the following protocol. (As stated before, we also modify BasicCLP so that P gives N_{col} parallel commitments in each slot.)

Protocol OurZK

Stage 1. V chooses a hash function $h \in \mathcal{H}_n$ and sends it to P.

Stage 2. For each $i \in [N_{\mathrm{slot}}]$ in sequence, P and V do the following.

Π-slot: P sends $C_{i,1} \leftarrow \mathsf{Com}(0^n), \ldots, C_{i,N_{\mathrm{col}}} \leftarrow \mathsf{Com}(0^n)$ to V. Then, V sends a random string $r_i \in \{0,1\}^{n^2}$ to P.

UA-slot: P sends $D_{i,1} \leftarrow \mathsf{Com}(0^n), \ldots, D_{i,N_{\mathrm{col}}} \leftarrow \mathsf{Com}(0^n)$ to V. Then, V sends a random string ω_i to P.

Stage 3. P proves the following statement with WIPOK.

- $x \in L$, or
- there exist $i_1, i_2 \in [N_{\mathrm{slot}}]$, $j \in [N_{\mathrm{col}}]$, and a second- and a fourth-round UA message UA_2 and UA_4 such that $D_{i_2,j}$ is a commitment to UA_2 and $(h, \mathsf{UA}_2, \omega_{i_2}, \mathsf{UA}_4)$ is an accepting proof of the statement $(h, C_{i_1,j}, r_{i_1}) \in \Lambda$.

We then consider the following simulator. Recall that, as explained above, our simulator commits to machines that emulate the simulation from the beginning of the current blocks, and its running time can be bounded by a polynomial if the offline proofs are computed only at the end of the blocks. Recall also that we need to make sure that (i) in each block the simulator uses only the offline proofs that are generated in that block (so

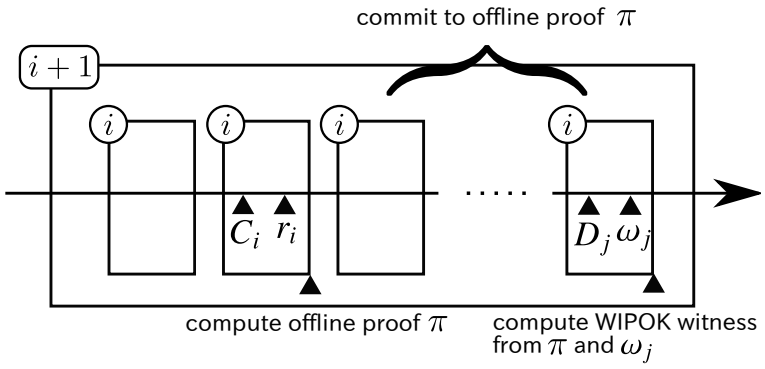

Figure 5.3: Our simulator's strategy.

that each committed machine does not output fail due to the lack of the offline proofs), and (ii) the simulator does not get stuck.

Roughly speaking, our simulator does the following in each block (see Figure 5.3). Consider any level-$(i+1)$ block b ($i \in [d-1]$), and recall that b consists of q level-i blocks. The goal of our simulator in block b is to compute offline proofs by using the machines that emulate the simulation from the beginning of those level-i blocks, and find opportunities to use them before block b completes. Therefore, for each session s, our simulator first tries to find a level-i block that contains a Π-slot of session s. If it finds such a level-i block and a Π-slot, it computes an offline proof at the end of that level-i block by using the machine that is committed to in the i-th column of that Π-slot, and commits to this offline proof in the i-th column of the UA-slots of session s in the subsequent level-i blocks. If a subsequent level-i block contains a UA-slot of session s, it computes a WIPOK witness from this offline proof (i.e., by using the third-round UA message in that UA-slot, it computes a fourth-round UA message from that offline proof).

More precisely, we consider the following simulator. In what follows, for each $i \in \{0, \ldots, d-1\}$, we say that two level-i blocks are *sibling* if they are contained by the same level-$(i+1)$ block.

- In the i-th column ($i \in [d]$) of a Π-slot of a session s, our simulator commits to a machine that emulates the simulator from the beginning of the current level-i block.

- In the i-th column ($i \in [d]$) of a UA-slot of a session s, our simulator commits to 0^n if no prior sibling of the current level-i block contains a Π-slot of session s; if a prior sibling contains a Π-slot of session s, an offline proof w.r.t. such a Π-slot was computed at the end of that prior sibling (see below), so our simulator commits to that offline proof instead of 0^n.

- When WIPOK starts, our simulator does the following. If it already obtained a valid witness (see below), it gives a proof by using this witness. If it does not have a valid witness, it aborts with output stuck.

- At the end of a level-i block b ($i \in [d-1]$), our simulator does the following. For each Π-slot that is contained in block b, it computes an offline proof by using

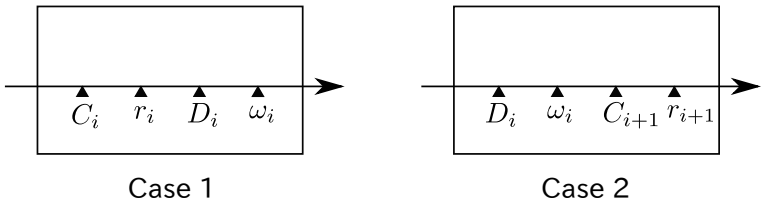

Figure 5.4: If a block contains two slots of a session, it contains both a Π-slot and a UA-slot.

the machine that is committed to in the i-th column of that Π-slot. Also, for each UA-slot that is contained in block b, if an offline proof is committed to in the i-th column of that UA-slot, it computes a WIPOK witness by using that offline proof.

In the simulation by our simulator, the committed machines never fail due to the lack of the offline proofs. This is because in each block, our simulator uses only the offline proofs that are generated in that block.

Thus, it remains to show that our simulator does not get stuck, i.e., in each session our simulator has a valid witness when WIPOK starts. Below, we use the following terminologies.

- For any session s, a block is *good* w.r.t. s if it contains at least two slots of session s and does not contain the first prover message of WIPOK of session s. Here, we use "slots" to refer to both Π-slots and UA-slots. Hence, if a block is good w.r.t. session s, it contains both a Π-slot and a UA-slot of session s (see Figure 5.4).

- For each $i \in [d]$, we say that a level-$(i-1)$ block is a *child* of a level-i block if the former is contained by the latter. (Thus, each block has q children.)

From the construction, our simulator does not get stuck if for any session s that reaches WIPOK, there exists a block such that at least two of its children are good w.r.t. session s. (Indeed, if such a block exists, an offline proof is computed at the end of the first good child, and a WIPOK witness is computed at the end of the second good child, so the simulator obtains a WIPOK witness before WIPOK starts in session s.) Thus, we show that if a session s reaches WIPOK, there exists a block such that at least two of its children are good w.r.t. session s. To prove this, it suffices to show that if a session s reaches WIPOK, there exists a block such that at least three of its children contain two or more slots of session s. (This is because at most one child contains the first message of WIPOK of session s.) Assume for contradiction that there exists a session s^* such that s^* reaches WIPOK but every block has at most two children that contain two or more slots of s^*. Let $\Gamma(i)$ be the maximum number of the slots that belong to s^* and are contained by a level-i block. Then, since in each block b,

- at most two children of b contain two or more slots of s^*, and the other children contain at most a single slot of s^*, and

- s^* has at most $q-1$ slots that are contained by block b but are not contained by its children (see Figure 5.5),

we have

$$\Gamma(i) \leq 2 \cdot \Gamma(i-1) + (q-2) \cdot 1 + q - 1 = 2\Gamma(i-1) + 2q - 3 \ .$$

Then, since $\Gamma(0) = 0$ (this is because a level-0 block contains only a single message), and the maximum level d is constant, we have

$$\Gamma(d) \leq 2^d \Gamma(0) + \sum_{i=0}^{d-1} 2^i (2q-3) = O(q) \ .$$

This means that there are at most $O(q)$ slots of s^* in the entire transcript. This is a contradiction since we have $N_{\mathrm{slot}} = \omega(q)$ and assume that s^* reaches WIPOK. Thus, if a session reaches WIPOK, there exists a block such that at least two of its children are good w.r.t. that session. Thus, the simulator does not get stuck.

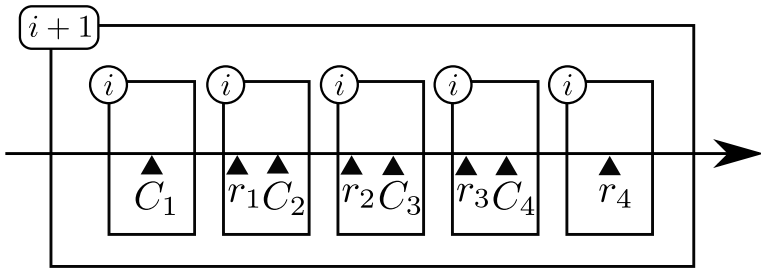

Figure 5.5: An example that a session has $q-1$ slots that are contained by a block but are not contained by its children. (For simplicity, only Π-slots are illustrated.)

Since $q = O(n^\epsilon)$ and $N_{\mathrm{slot}} = \omega(q)$, the round complexity of our protocol is $O(n^{\epsilon'})$ for a constant $\epsilon' > \epsilon$. Since ϵ is an arbitrary constant, ϵ' can be an arbitrary small constant.

Toward the final protocol. To obtain a formal proof of security, we need to add a slight modification to the above protocol. Specifically, as pointed out in previous work [Goy13, CLP13a, CLP13b, PPS15], when the code of the simulator is committed in the simulation, we have to take special care to the randomness of the simulator.[36] Fortunately, the techniques used in the previous work can also be used here to overcome this problem. In this work, we use the technique of [CLP13a, CLP13b], which uses forward-secure pseudorandom generators (which can be obtained from one-way functions).

[36] When the code of the simulator is committed, the randomness used for generating this commitment is also committed; thus, if a protocol is designed naively, we need a commitment scheme such that the committed value is hidden even when it contains the randomness used for the commitment.

5.2.3 Comparison with the Non-black-box Simulation Technique of Goyal [Goy13]

In this section, we compare the simulation technique of ours with that of Goyal [Goy13], which is the only known simulation technique that realizes straight-line concurrent simulation in the plain model under standard assumptions.

First of all, our protocol is almost identical with that of Goyal. The only difference is that the prover gives $\omega(1)$ commitments in each slot in our protocol whereas it gives only a single commitment in each slot in Goyal's protocol.

Our simulation technique is also very similar to Goyal's simulation technique. For example, in both simulation techniques, the simulator commits to machines that emulate itself, and it has multiple opportunities to give UA proof and determines which opportunities to take by using the blocks.

However, there are also differences between the two simulation techniques. A notable difference is how the simulator determines which opportunities to take to give UA proofs. Recall that, in the simulation technique of ours, the strategy that the simulator uses to determine whether it embeds a UA message in a slot is deterministic (the simulator checks whether a prior sibling of the current block contains a Π-slot; see Figure 5.3 in Section 5.2.2). In contrast, in the simulation technique of Goyal, the strategy that the simulator uses is probabilistic (the simulator uses a probabilistic procedure that performs the "marking" of the blocks and the UA messages). Since in the simulation technique of ours the simulator uses a deterministic strategy, the analysis of our simulator is simple: We use only a simple counting argument (and no probabilistic argument) to show that the simulator will not get stuck.

5.3 Preliminaries

5.3.1 Notations

We use Com to denote Naor's 2-round statistically binding commitment scheme, and $\mathsf{Com}_\tau(\cdot)$ to denote an algorithm that, on input $m \in \{0,1\}^*$, computes a commitment to m by using Naor's commitment scheme with the first-round message being τ (cf. Section 2.3.1).

5.3.2 Tree Hashing

In this chapter, we use a family of collision-resistant hash functions $\mathcal{H} = \{h_\alpha\}_{\alpha\in\{0,1\}^*}$ that satisfies the following properties.

- For any $h \in \mathcal{H}_n \stackrel{\text{def}}{=} \{h_\alpha \in \mathcal{H} : \alpha \in \{0,1\}^n\}$, the domain of h is $\{0,1\}^*$ and the range of h is $\{0,1\}^n$.
- For any $h \in \mathcal{H}_n$, $x \in \{0,1\}^{\leq n^{\log\log n}}$, and $i \in \{1,\ldots,|x|\}$, one can compute a short certificate $\mathsf{auth}_i(x) \in \{0,1\}^{n^2}$ such that given $h(x)$, x_i, and $\mathsf{auth}_i(x)$, anyone can verify that the i-th bit of x is indeed x_i.

We obtain such a collision-resistant hash function family from any (standard) length-halving collision-resistant hash function family by using Merkle's tree-hashing technique. We notice that when $\mathcal{H}$ is obtained in this way, $\mathcal{H}$ satisfies an additional property that one can find a collision of the underlying hash function from any two "contradicting" certificates, i.e., two pairs $(x_i, \mathsf{auth}_i(x))$ and $(x'_i, \mathsf{auth}_i(x'))$ such that $x_i \neq x'_i$, and finding a collision in this way takes only time polynomial in the size of the hash value (i.e., $|h(x)| = n$).

5.3.3 Concurrent Zero-Knowledge Arguments

In this section, we recall the definition of the *concurrent zero-knowledge* property of interactive proofs and arguments from [RK99]. For any polynomial $m(\cdot)$, *m-session concurrent cheating verifier* is a PPT Turing machine V^* such that on input (x, z), V^* concurrently interacts with $m(|x|)$ independent copies of P. The interaction between V^* and each copy of P is called *session*. There is no restriction on how V^* schedules messages among sessions, and V^* can abort some sessions. Let $\mathsf{view}_{V^*}[P(x, w) \leftrightarrow V^*(x, z)]$ be the view of V^* in the above concurrent execution, where $x \in L$ is the common input, $w \in \mathbf{R}_L(x)$ is the private input to P, and z is the non-uniform input to V^*.

Definition 5.1 (Concurrent Zero-Knowledge)**.** *An interactive proof (or argument) $\langle P, V\rangle$ for an $\mathcal{NP}$ language L is **concurrent zero-knowledge** if for every polynomial $m(\cdot)$ and every m-session concurrent cheating verifier V^*, there exists a PPT simulator $\mathcal{S}$ such that for any sequence $\{w_x\}_{x\in L}$ such that $w_x \in \boldsymbol{R}_L(x)$, the following indistinguishability holds.*

$$\{\mathsf{view}_{V^*}[P(x, w_x) \leftrightarrow V^*(x, z)]\}_{x\in L, z\in\{0,1\}^*} \overset{c}{\approx} \{\mathcal{S}(x, z)\}_{x\in L, z\in\{0,1\}^*} \quad .$$

◇

Remark 5.1. As in previous work (e.g., [RK99, KP01, PRS02]), we consider the setting where the same statement x is proven in all the sessions. We comment that our protocol and its security proof work even in a slightly generalized setting where predetermined statements $x_1, \ldots, x_m$ are proven in the sessions. (However, they do not work if the statements are chosen adaptively by the cheating verifier.)

5.3.4 PCP and Universal Argument

In this section, we explain the universal arguments system that we use in this chapter, namely the universal argument system of Barak and Goldreich [BG08]. (Recall that the definition of universal arguments is given in Section 2.5.) Since their instantiation uses *probabilistically checkable proof* (PCP) systems [AS98] as a building block, we also recall the definition of PCP systems below.

Recall that, as noted in Section 2.5, universal arguments are used in this thesis only for proving the membership of a single "universal" language $L_{\mathcal{U}}$, where for any triplet $y = (M, x, t)$, we have $y \in L_{\mathcal{U}}$ if non-deterministic machine M accepts x within t steps.

5.3.4.1 PCP System

Roughly speaking, a PCP system is a PPT verifier that can decide the correctness of a statement $y \in L_{\mathcal{U}}$ given access to an oracle π that represents a proof in a redundant form. Typically, the verifier reads only few bits of π in the verification.

Definition 5.2 (PCP system—basic definition)**.** *A **probabilistically checkable proof** (PCP) system (with a negligible soundness error) is a PPT oracle machine V (called a verifier) that satisfies the following.*

- ***Completeness:*** *For every $y \in L_{\mathcal{U}}$, there exists an oracle π such that*

$$\Pr\left[V^{\pi}(y) = 1\right] = 1 \quad .$$

- ***Soundness:*** *For every $y \notin L_{\mathcal{U}}$ and every oracle π, there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\Pr\left[V^{\pi}(y) = 1\right] < \mathsf{negl}(|y|) \quad .$$

◇

In this thesis, PCP systems are used as a building block in the universal argument system of Barak and Goldreich [BG08]. To be used in their universal argument system, PCP systems need to satisfy four auxiliary properties: relatively efficient oracle construction, non-adaptive verifier, efficient reverse sampling, and proof of knowledge. Only the definitions of the first two properties are required to understand this chapter; for the definitions of the other properties, see [BG08].

Definition 5.3 (PCP system—auxiliary properties)**.** *Let V be a PCP-verifier.*

- ***Relatively efficient oracle construction:*** *There exists an algorithm P (called a prover) such that, given any $(y, w) \in \boldsymbol{R}_{\mathcal{U}}$, algorithm P outputs an oracle π_y that makes V always accept (i.e., as in the completeness condition). Furthermore, there exists a polynomial $p(\cdot)$ such that on input (y, w), the running time of P is $p(|y| + |w|)$.*

- ***Non-adaptive verifier:*** *The verifier's queries are determined based only on the input and its internal coin tosses, independently of the answers given to previous queries. That is, V can be decomposed into a pair of algorithms Q and D such that on input y and random tape r, the verifier makes the query sequence $Q(y, r, 1), Q(y, r, 2), \ldots, Q(y, r, p(|y|))$, obtains the answers $b_1, \ldots, b_{p(|y|)}$, and decides according to $D(y, r, b_1 \cdots b_{p(|y|)})$, where p is some fixed polynomial.*

◇

5.3.4.2 Universal Argument System of [BG08]

The public-coin four-round universal argument system UA of Barak and Goldreich [BG08] is shown in Figure 5.6. As in [CLP13a], the construction of UA is separated into an *offline stage* and an *online stage*. In the offline stage, the running time of the prover is bounded by a fixed polynomial in $n + T_M(x, w)$. (Recall that for any $y = (M, x, t) \in L_{\mathcal{U}}$, we use $T_M(x, w)$ to denote the running time of M on input x with witness w; cf. Section 2.5.)

- **Input:** The common input is $y = (M, x, t) \in L_{\mathcal{U}}$, and the private input to P is $w \in \mathbf{R}_{\mathcal{U}}(y)$. Let $n \stackrel{\text{def}}{=} |y|$.
- **Offline Phase:**
  1. V sends a random hash function $h \in \mathcal{H}_n$ to P.
  2. P generates a PCP proof π of statement $y \in L_{\mathcal{U}}$ by using w as a witness. Then P computes $\mathsf{UA}_2 := h(\pi)$. The tuple (h, π, UA_2) is called the *offline proof*.
- **Online Phase:**
  1. P sends UA_2 to V.
  2. V chooses randomness $\omega \in \{0,1\}^{n^2}$ for the PCP-verifier and sends $\mathsf{UA}_3 := \omega$ to P.
  3. P computes queries Q by executing the PCP-verifier with statement $y \in L_{\mathcal{U}}$ and randomness ω. Then, P sends $\mathsf{UA}_4 := \{(i, \pi_i, \mathsf{auth}_i(\pi))\}_{i \in Q}$ to V, where π_i is the i-th bit of π and $\mathsf{auth}_i(\pi)$ is a certificate that the i-th bit of π is indeed π_i.
  4. V verifies the correctness of all the certificates and checks whether the PCP-verifier accepts on input $(y, \{(i, \pi_i)\}_{i \in Q})$ with randomness ω.

Figure 5.6: Online/offline UA system of [BG08, CLP13a].

5.3.5 Forward-secure PRG

In this section, we recall the definition of *forward-secure pseudorandom generators* (PRGs) [BY03]. Roughly speaking, a forward-secure PRG is a pseudorandom generator such that

- It periodically updates the seed. Hence, we have a sequence of seeds $(\sigma_1, \sigma_2, \ldots)$ that generates a sequence of pseudorandomness $(\rho_1, \rho_2, \ldots)$.
- Even if the seed σ_t is exposed (and thus the "later" pseudorandom sequence $\rho_{t+1}, \rho_{t+2}, \ldots$ is also exposed), the "earlier" sequence $\rho_1, \ldots, \rho_t$ remains pseudorandom.

In this chapter, we use a simple variant of the definition by [CLP13b]. We notice that in the following definition, the indices of the seeds and pseudorandomness are written in the reverse order because we use them in the reverse order in the analysis of our concurrent zero-knowledge protocol.

Definition 5.4 (Forward-secure PRG)**.** *We say that a polynomial-time computable function* $\mathsf{f\text{-}PRG}$ *is a* ***forward-secure pseudorandom generator*** *if on input a string* σ *and an integer* $\ell \in \mathbb{N}$*, it outputs two sequences* $(\sigma_\ell, \ldots, \sigma_1)$ *and* $(\rho_\ell, \ldots, \rho_1)$ *that satisfy the following properties.*

- ***Consistency:*** *For every* $n, \ell \in \mathbb{N}$ *and* $\sigma \in \{0,1\}^n$*, if* $\mathsf{f\text{-}PRG}(\sigma, \ell) = (\sigma_\ell, \ldots, \sigma_1, \rho_\ell, \ldots, \rho_1)$*, then it holds* $\mathsf{f\text{-}PRG}(\sigma_\ell, \ell - 1) = (\sigma_{\ell-1}, \ldots, \sigma_1, \rho_{\ell-1}, \ldots, \rho_1)$*.*

- ***Forward Security:*** *For every polynomial* $\ell(\cdot)$*, the following ensembles are computationally indistinguishable.*

  - $\left\{(\sigma_{\ell(n)}, \rho_{\ell(n)}) \;\middle|\; \sigma \leftarrow U_n; (\sigma_{\ell(n)}, \ldots, \sigma_1, \rho_{\ell(n)}, \ldots, \rho_1) := \mathsf{f\text{-}PRG}(\sigma, \ell(n))\right\}_{n\in\mathbb{N}}$
  - $\left\{(\sigma, \rho) \;\middle|\; \sigma \leftarrow U_n; \rho \leftarrow U_n\right\}_{n\in\mathbb{N}}$

  Here, U_n *is the uniform distribution over* $\{0,1\}^n$*.* ◇

The existence of (standard) PRGs implies the existence of forward-secure PRGs. Thus from the result of [HILL99], the existence of forward-secure PRGs is implied by the existence of one-way functions.

5.4 Our Public-Coin Concurrent Zero-Knowledge Argument

In this section, we prove our main theorem.

Theorem 5.1 (restatement of Main Theorem)**.** *Assume the existence of a family of collision resistant hash functions. Then, for any constant* $\epsilon > 0$*, there exists an* $O(n^\epsilon)$*-round public-coin concurrent zero-knowledge argument of knowledge system.*

Proof. Our $O(n^\epsilon)$-round public-coin concurrent zero-knowledge argument of knowledge, cZKAOK, is shown in Figure 5.7. In cZKAOK, we use the following building blocks.

- Naor's two-round statistically binding commitment scheme Com, which can be constructed from one-way functions (see Section 2.3.1).

- A four-round public-coin witness-indistinguishable proof of knowledge system WIPOK, which can be constructed from one-way functions by instantiating a parallel version of Blum's Hamiltonian-cycle protocol with Naor's commitment scheme (see Section 2.4.2) .

- Four-round public-coin universal argument UA of Barak and Goldreich [BG08], which can be constructed from collision-resistant hash functions (see Section 5.3.4.2).

Clearly, cZKAOK is public-coin and its round complexity is $4N_{\mathrm{slot}} + 5 = O(n^\epsilon)$. Thus, Theorem 5.1 follows from the following lemmas.

Lemma 5.1. cZKAOK *is concurrent zero-knowledge.*

Lemma 5.2. cZKAOK *is argument of knowledge.*

Lemma 5.1 is proven in Section 5.4.1, and Lemma 5.2 is proven in Section 5.4.2. □

Input: The input to the prover P is (x, w), where $x \in L$ and $w \in \mathbf{R}_L(x)$. The input to the verifier V is x. Let $n \stackrel{\text{def}}{=} |x|$.

Parameter: Integers $N_{\text{slot}} = O(n^{\epsilon})$ and $N_{\text{col}} = \omega(1)$, where ϵ is an arbitrary constant.

Stage 1: V chooses a random hash function $h \in \mathcal{H}_n$ and sends it to P. Additionally, V sends a first-round message $\tau \in \{0,1\}^{3n}$ of Com to P.

Stage 2: For each $i \in [N_{\text{slot}}]$ in sequence, P and V do the following.

Π-slot:

1. P computes $C_{i,j} \leftarrow \mathsf{Com}_\tau(0^n)$ for each $j \in [N_{\text{col}}]$. Then, P sends $\overline{C}_i := (C_{i,1}, \ldots, C_{i,N_{\text{col}}})$ to V.
2. V sends random $r_i \in \{0,1\}^{n^2}$ to P.

UA-slot:

3. P computes $D_{i,j} \leftarrow \mathsf{Com}_\tau(0^n)$ for each $j \in [N_{\text{col}}]$. Then, P sends $\overline{D}_i := (D_{i,1}, \ldots, D_{i,N_{\text{col}}})$ to V.
4. V computes a third-round message of UA. (Recall that a third-round message of UA is just a random string with appropriate length.) Then, V sends this UA message, ω_i, to P.

Stage 3: P proves the following statement by using WIPOK.

- $x \in L$, or
- $(h, \tau, \overline{C}_1, r_1, \overline{D}_1, \omega_1, \ldots, \overline{C}_{N_{\text{slot}}}, r_{N_{\text{slot}}}, \overline{D}_{N_{\text{slot}}}, \omega_{N_{\text{slot}}}) \in \Lambda_1$, where the language Λ_1 is defined in Figure 5.8.

Figure 5.7: Public-coin concurrent zero-knowledge argument cZKAOK.

Remark 5.2. The languages Λ_2 in Figure 5.8 is slightly over-simplified and will make cZKAOK work only when $\mathcal{H}$ is collision resistant against $\mathsf{poly}(n^{\log\log n})$-time adversaries. We can make it work under standard collision resistance by using a trick by Barak and Goldreich [BG08], which uses a "good" error-correcting code ECC (i.e., with constant relative distance and with polynomial-time encoding and decoding). More details are given in Section 5.4.2.

5.4.1 Concurrent Zero-knowledge Property

Proof of Lemma 5.1. Let V^* be any cheating verifier. Since V^* takes an arbitrary non-uniform input z, we assume without loss of generality that V^* is deterministic. Let $m(\cdot)$ be a polynomial such that V^* invokes $m(n)$ concurrent sessions during its execution. (Recall that $n \stackrel{\text{def}}{=} |x|$.) Let $q \stackrel{\text{def}}{=} n^{\epsilon/2}$. We assume without loss of generality that in the interaction between V^* and provers, the total number of messages across all the

Language Λ_1: (Statement for WIPOK)

$(h, \tau, \overline{C}_1, r_1, \overline{D}_1, \omega_1, \ldots, \overline{C}_{N_{\mathrm{slot}}}, r_{N_{\mathrm{slot}}}, \overline{D}_{N_{\mathrm{slot}}}, \omega_{N_{\mathrm{slot}}}) \in \Lambda_1$ if and only if there exist

- $i_1, i_2 \in [N_{\mathrm{slot}}]$ and $j \in [N_{\mathrm{col}}]$ such that $i_1 \leq i_2$
- a second- and a fourth-round UA message $\mathsf{UA}_2 \in \{0,1\}^n$ and $\mathsf{UA}_4 \in \{0,1\}^{\mathsf{poly}(n)}$
- randomness $R \in \{0,1\}^{\mathsf{poly}(n)}$ for Com

such that

- $D_{i_2,j} = \mathsf{Com}_\tau(\mathsf{UA}_2; R)$, and
- $(h, \mathsf{UA}_2, \omega_{i_2}, \mathsf{UA}_4)$ is an accepting proof for $(h, \tau, C_{i_1,j}, r_{i_1}) \in \Lambda_2$.

Language Λ_2: (Statement for UA)

$(h, \tau, C, r) \in \Lambda_2$ if and only if there exist

- a machine Π (with some of the inputs being hardwired) such that $|\Pi| \leq n^{\log\log n}$
- randomness $R \in \{0,1\}^{\mathsf{poly}(n)}$ for Com
- a string y such that $|y| = n$

such that

- $C = \mathsf{Com}_\tau(h(\Pi); R)$, and
- $\Pi(y)$ outputs a string that has r as a substring, and $\Pi(y)$ outputs it within $n^{\log\log n}$ steps.

Figure 5.8: Languages used in cZKAOK.

sessions is always the power of q (i.e., it is q^d for an integer d). Since the total number of messages is at most $M \stackrel{\mathrm{def}}{=} (4N_{\mathrm{slot}}+5)\cdot m$, we have $d = \log_q M = \log_q(\mathsf{poly}(n)) = O(1)$.

5.4.1.1 Simulator $\mathcal{S}$

In this section, we describe our simulator. We first give an informal description; a formal description is given after the informal one. We recommend the readers to browse the overview of our techniques in Section 5.2.2 before reading this section. In the informal description, we use some terminologies that we introduced in Section 5.2.2.

Informal Description of $\mathcal{S}$

Our simulator, $\mathcal{S}$, simulates the view of V^* by using an auxiliary simulator algorithm $\mathsf{aux}\text{-}\mathcal{S}$, which simulates the transcript between V^* and honest provers by recursively

executing itself. The input to aux-$\mathcal{S}$ is the recursion level ℓ and the transcript trans that is simulated so far. aux-$\mathcal{S}$ is also given oracle access to tables $\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$ (which aux-$\mathcal{S}$ can freely read and update), where T_Π contains the hash values of the machines that should be committed to in the Π-slots, and T_{UA} and T_{W} contain the second-round UA messages and the WIPOK witnesses that are computed so far. The goal of aux-$\mathcal{S}$, on input (ℓ, trans) and access to $\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$, is to add the next q^ℓ messages to trans while updating the tables $\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$. More details about aux-$\mathcal{S}$ are described below.

On level 0 (i.e., when $\ell = 0$), aux-$\mathcal{S}$ adds a single message to the simulated transcript as follows. If the next message is a verifier message, aux-$\mathcal{S}$ simulates it by simply receiving it from V^*. If the next message is a prover message $(C_1, \ldots, C_{N_{\mathrm{col}}})$ in a Π-slot, aux-$\mathcal{S}$ finds the values to be committed from T_Π and generates commitments to them by using Com. Similarly, if the next message is a prover message $(D_1, \ldots, D_{N_{\mathrm{col}}})$ in a UA-slot, aux-$\mathcal{S}$ finds appropriate second-round UA messages from T_{UA} and generates commitments to them by using Com. (If appropriate UA messages cannot be found, aux-$\mathcal{S}$ generates commitments to 0^n.) If the next message is a prover message of WIPOK, aux-$\mathcal{S}$ computes it honestly by using a witness that is stored in T_{W}. (If the stored witness is not a valid witness, aux-$\mathcal{S}$ aborts.)

On level $\ell > 0$, aux-$\mathcal{S}$ simulates the next q^ℓ messages by recursively executing itself q times in sequence, where each recursive execution simulates $q^{\ell-1}$ messages. More precisely, aux-$\mathcal{S}$ first updates T_Π by storing the hash values of its own code (with the inputs and the entries of the tables being hardwired), where the hash functions of all the existing sessions are used for computing these hash values, and each hash value is stored as the value to be committed in the ℓ-th commitment in Π-slots. (By requiring aux-$\mathcal{S}$ to store its own code in T_Π in this way, we make sure that when aux-$\mathcal{S}$ simulates a Π-slot, it commits to its own code in the Π-slot.) Then, aux-$\mathcal{S}$ recursively executes itself q times in sequence with level $\ell - 1$; at the same time, aux-$\mathcal{S}$ updates $\mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$ at the end of each recursive execution in the following way.

- If a Π-slot (both the prover message and the verifier message) of a session is simulated by the recursive execution that has just been completed, aux-$\mathcal{S}$ computes a second-round UA message about such a Π-slot and stores it in T_{UA}.

  (A machine that emulates this recursive execution must be committed in the $(\ell - 1)$-th commitment of such a Π-slot (this is because the recursively executed aux-$\mathcal{S}$ must have stored its own code in T_Π at the beginning of its execution), and this machine can be used as a witness for generating a UA proof about this Π-slot.)

- If a UA-slot of a session is simulated by the recursive execution that has just been completed, and a second-round UA message for this session was stored in T_{UA} before this recursive execution, aux-$\mathcal{S}$ computes a WIPOK witness for this session and stores it in T_{W}.

  (If such a UA-slot and a second-round UA message exist, that second-round UA message must be committed to in that UA-slot, and they can be used as a WIPOK witness.)

Finally, aux-$\mathcal{S}$ outputs the q^ℓ messages that are simulated by these q recursive executions.

We remark that for technical reasons, the formal description of $\mathcal{S}$ below is a bit more complex.

- To avoid the circularity that arises when aux-$\mathcal{S}$ uses its own code, we use a technique by Chung, Lin, and Pass [CLP13b]. Roughly speaking, aux-$\mathcal{S}$ takes the code of a machine Π as input and uses this code rather than its own code; we then design $\mathcal{S}$ and aux-$\mathcal{S}$ in such a way that when aux-$\mathcal{S}$ is invoked, we always have Π = aux-$\mathcal{S}$.

- To avoid the circularity issue about randomness that we sketched in Section 5.2.2, we use a technique of [CLP13a, CLP13b] that uses a forward-secure PRG f-PRG. Roughly speaking, aux-$\mathcal{S}$ takes a seed σ of f-PRG as input, computes a sequence of pseudorandomness $\rho_{q^\ell}, \ldots, \rho_2, \rho_1$ (notice that the indices are written in the reverse order), and simulates the prover messages in such a way that the i-th message in the transcript is simulated with randomness ρ_i.

Formal Description of $\mathcal{S}$

The input to $\mathcal{S}$ is (x, z), and the input to aux-$\mathcal{S}$ is $(x, z, \ell, \Pi, \mathsf{trans}, \sigma)$ such that:

- $\ell \in \{0, \ldots, d\}$.
- Π is a code of a machine. (In what follows, we always have Π = aux-$\mathcal{S}$.)
- $\mathsf{trans} \in \{0,1\}^{\mathsf{poly}(n)}$ is a prefix of a transcript between $V^*(x, z)$ and honest provers.
- $\sigma \in \{0,1\}^n$ is a seed of f-PRG.

The auxiliary simulator aux-$\mathcal{S}$ is also given oracle access to three tables $\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$ such that:

- $\mathsf{T}_\Pi = \{v_{s,j}\}_{s\in[m], j\in[N_{\mathrm{col}}]}$ is a table of the hash values of machines.
- $\mathsf{T}_{\mathsf{UA}} = \{\mathsf{ua}_{s,j}\}_{s\in[m], j\in[N_{\mathrm{col}}]}$ is a table of second-round UA messages.
- $\mathsf{T}_{\mathsf{W}} = \{w_s\}_{s\in[m]}$ is a table of WIPOK witnesses.

We allow aux-$\mathcal{S}$ to read and update the entries in $\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$ freely.

Simulator $\mathcal{S}(x, z)$:

1. Choose a random seed $\sigma_{q^d+1} \in \{0,1\}^n$ of f-PRG. Initialize $\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$ by $v_{s,j} := 0^n$, $\mathsf{ua}_{s,j} := 0^n$, $w_s := \bot$ for every $s \in [m]$ and $j \in [N_{\mathrm{col}}]$.
2. Compute $\mathsf{trans} := \text{aux-}\mathcal{S}^{\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}}(x, z, d, \text{aux-}\mathcal{S}, \varepsilon, \sigma_{q^d+1})$, where ε is the empty string.
3. Output (x, z, trans).

Auxiliary Simulator aux-$\mathcal{S}^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x,z,\ell,\Pi,\mathsf{trans},\sigma)$**:**

Step 1.
`/* Preparing randomness for simulating the next` q^ℓ `messages */`
Let $\kappa := |\mathsf{trans}|$ (i.e., κ be the number of the messages that are included in trans). Then, compute

$$(\sigma_{\kappa+q^\ell},\ldots,\sigma_1,\rho_{\kappa+q^\ell},\ldots,\rho_1) := \mathsf{f\text{-}PRG}(\sigma,\kappa+q^\ell) \ .$$

Step 2a: Simulation (base case). If $\ell = 0$, do the following.

1. If the next-scheduled message msg is a verifier message, feed trans to $V^*(x,z)$ and receive msg from V^*.

   If the next-scheduled message msg is a prover message of the s-th session ($s \in [m]$), do the following with randomness $\rho_{\kappa+1}$. (If necessary, $\rho_{\kappa+1}$ is expanded by a pseudorandom generator.) Let τ_s be the first-round message of Com of the s-th session, which can be found from trans.

   - If msg is the prover message in a Π-slot, compute $\mathsf{msg} = (C_1,\ldots,C_{N_{\mathrm{col}}})$ by $C_j \leftarrow \mathsf{Com}_{\tau_s}(v_{s,j})$ for every $j \in [N_{\mathrm{col}}]$, where $v_{s,j}$ is read from T_Π.
   - If msg is the prover message in a UA-slot, compute $\mathsf{msg} = (D_1,\ldots,D_{N_{\mathrm{col}}})$ by $D_j \leftarrow \mathsf{Com}_{\tau_s}(\mathsf{ua}_{s,j})$ for every $j \in [N_{\mathrm{col}}]$, where $\mathsf{ua}_{s,j}$ is read from T_{UA}.
   - If msg is the first prover message of WIPOK, compute msg by using w_s as a witness, where w_s is read from T_{W}; if w_s is not a valid witness, abort with output stuck.
   - If msg is the second prover message of WIPOK, reconstruct the prover state of WIPOK from w_s and $\rho_1,\ldots,\rho_\kappa$ and then honestly compute msg by using this prover state, where w_s is read from T_{W}.[37]

2. Output msg.

Step 2b: Simulation (recursive case). If $\ell > 0$, do the following.

1. `/* Storing its own code in` T_Π `*/`
   Let T'_Π, $\mathsf{T}'_{\mathsf{UA}}$, T'_{W} be the tables that are obtained by copying the current entries of T_Π, T_{UA}, T_{W}. Let

   $$\Pi_{\mathrm{myself}}(\cdot) \stackrel{\mathrm{def}}{=} \Pi^{\mathsf{T}'_\Pi,\mathsf{T}'_{\mathsf{UA}},\mathsf{T}'_{\mathsf{W}}}(x,z,\ell,\Pi,\mathsf{trans},\cdot) \ .$$

   Then, for each $s \in [m]$, if the s-th session has already started in trans, do the following: Let h_s be the hash function chosen by V^* in the s-th session in trans; then, update T_Π by setting $v_{s,\ell} := h_s(\Pi_{\mathrm{myself}})$.

[37] From the construction of $\mathcal{S}$ and aux-$\mathcal{S}$, the first prover message of WIPOK in trans must have been computed with witness w_s and randomness in $\rho_1,\ldots,\rho_\kappa$.

2. Set temporary variables $\mathsf{ctr}_s := 0$ and $\mathsf{tmp}_s := \bot$ for every $s \in [m]$, and a temporary variable $\mathsf{new\text{-}trans} := \varepsilon$. Define a function $\mathsf{head}(\cdot)$ as $\mathsf{head}(k) \stackrel{\text{def}}{=} \kappa + 1 + (k-1)q^{\ell-1}$.

3. For each $k \in [q]$, do the following:

   (a) `/* Storing the machine that executes the k-th child-block. */`
   Let $\mathsf{T}_\Pi^{(k)}$, $\mathsf{T}_{\mathsf{UA}}^{(k)}$, $\mathsf{T}_{\mathsf{W}}^{(k)}$ be the tables that are obtained by copying the current entries of T_Π, T_{UA}, T_{W}. Then, let

   $$\Pi_k(\cdot) \stackrel{\text{def}}{=} \Pi^{\mathsf{T}_\Pi^{(k)},\mathsf{T}_{\mathsf{UA}}^{(k)},\mathsf{T}_{\mathsf{W}}^{(k)}}(x, z, \ell-1, \Pi, \mathsf{trans}\|\mathsf{new\text{-}trans}, \cdot) \ .$$

   (b) `/* Executing the k-th child-block. */`
   Compute

   $$\mathsf{trans}_k := \Pi^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x, z, \ell-1, \Pi, \mathsf{trans}\|\mathsf{new\text{-}trans}, \sigma_{\mathsf{head}(k+1)})$$

   while reading and updating T_Π, T_{UA}, T_{W} for Π.

   (c) For each $s \in [m]$, if the s-th session has already started in $\mathsf{trans}\|\mathsf{new\text{-}trans}$, do the following. Let (h_s, τ_s) be the first-round message of the s-th session.

   Case 1. $\mathsf{ctr}_s = 0$, and trans_k contains a Π-slot of the s-th session.
   `/* Computing offline proof */`
   Let sl denote the Π-slot that is contained by trans_k. (If there are more than one such Π-slot, sl denote the first such one.) Let i_1 denote the *slot-index* of sl, i.e., $i_1 \in [N_{\mathrm{slot}}]$ such that sl is the i_1-th Π-slot in the s-th session. Let $(\overline{C}_{i_1}, r_{i_1})$ denote the messages in sl, where $\overline{C}_{i_1} = (C_{i_1,1}, \ldots, C_{i_1,N_{\mathrm{col}}})$.

   i. From $\rho_{\mathsf{head}(k)}, \ldots, \rho_{\mathsf{head}(k+1)-1}$ (which are computed in Step 1), find the randomness R_1 that was used for generating $C_{i_1,\ell-1}$.[38] Then, compute a PCP proof π_s for statement $(h_s, \tau_s, C_{i_1,\ell-1}, r_{i_1}) \in \Lambda_2$ using $(\Pi_k, R_1, \sigma_{\mathsf{head}(k+1)})$ as a witness (see Section 5.4.1.2 for details). Then, compute $\mathsf{UA}_2 := h_s(\pi_s)$.

   ii. Update T_{UA} by setting $\mathsf{ua}_{s,\ell-1} := \mathsf{UA}_2$.

   iii. Update $\mathsf{tmp}_s := (i_1, \pi_s, \mathsf{UA}_2)$ and $\mathsf{ctr}_s := \mathsf{ctr}_s + 1$.

   Case 2. $\mathsf{ctr}_s = 1$, and trans_k contains a UA-slot of the s-th session.
   `/* Computing WIPOK witness */`
   Let sl denote the UA-slot that is contained by trans_k. (If there are more than one such UA-slot, sl denote the first such one.) Let i_2 be the slot-index of sl. Let $(\overline{D}_{i_2}, \omega_{i_2})$ denote the messages in sl, where $\overline{D}_{i_2} = (D_{i_2,1}, \ldots, D_{i_2,N_{\mathrm{col}}})$.

   i. Parse $(i_1, \pi_s, \mathsf{UA}_2) := \mathsf{tmp}_s$. Then, compute a fourth-round UA message UA_4 from the offline proof $(h_s, \pi_s, \mathsf{UA}_2)$ and the third-round UA message ω_{i_2}.

[38] From the construction of $\mathsf{aux}\text{-}\mathcal{S}$, every message in the k-th child-block is computed by using randomness in $\rho_{\mathsf{head}(k)}, \ldots, \rho_{\mathsf{head}(k+1)-1}$.

ii. From $\rho_{\mathsf{head}(k)}, \ldots, \rho_{\mathsf{head}(k+1)-1}$ (which are computed in Step 1), find the randomness R_2 that was used for generating $D_{i_2,\ell-1}$. Then, if $w_s = \bot$, update $\mathsf{T_W}$ by setting $w_s := (i_1, i_2, \ell - 1, \mathsf{UA}_2, \mathsf{UA}_4, R_2)$.

iii. Update $\mathsf{ctr}_s := \mathsf{ctr}_s + 1$.

Case 3. All the other cases.
Do nothing.

(d) Update $\mathsf{new\text{-}trans} := \mathsf{new\text{-}trans} \| \mathsf{trans}_k$.

4. Output new-trans.

5.4.1.2 Correctness of $\mathcal{S}$.

In this section, we observe the correctness of $\mathcal{S}$. Specifically, we observe that aux-$\mathcal{S}$ can indeed compute a valid PCP proof π_s and a valid WIPOK witness w_s in Step 2b.

First, we see that aux-$\mathcal{S}$ can indeed compute a PCP proof π_s in Step 2b. Specifically, we see that when aux-$\mathcal{S}$ computes π_s in Step 2b, $(\Pi_k, R_1, \sigma_{\mathsf{head}(k+1)})$ is indeed a witness for $(h_s, \tau_s, C_{i_1,\ell-1}, r_{i_1}) \in \Lambda_2$ (that is, $C_{i_1,\ell-1}$ is a commitment to $h_s(\Pi_k)$ and Π_k outputs a string that has r_{i_1} as a substring).

1. First, we observe that $C_{i_1,\ell-1}$ is a commitment to $h_s(\Pi_k)$. Recall that when aux-$\mathcal{S}$ computes π_s in Step 2b, the Π-slot $(C_{i_1,\ell-1}, r_{i_1})$ is contained by trans_k. Then, since trans_k is generated by a recursive execution

$$\Pi^{\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}}(x, z, \ell - 1, \Pi, \mathsf{trans}, \sigma_{\mathsf{head}(k+1)}) \ ,$$

and this recursive execution updates $v_{s,\ell-1} \in \mathsf{T}_\Pi$ to be the hash value of its own code at the beginning, $C_{i_1,\ell-1}$ is a commitment to the hash value of this code, which is identical with $h_s(\Pi_k)$.

2. Next, we observe that Π_k outputs trans_k on input $\sigma_{\mathsf{head}(k+1)}$. This is because from its definition $\Pi_k(\sigma_{\mathsf{head}(k+1)})$ is identical with

$$\Pi^{\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}}(x, z, \ell - 1, \Pi, \mathsf{trans}, \sigma_{\mathsf{head}(k+1)})$$

(including the entries in the tables), which outputs trans_k.

Since r_{i_1} is contained by trans_k, we conclude that $(\Pi_k, R_1, \sigma_{\mathsf{head}(k+1)})$ is indeed a witness for $(h_s, \tau_s, C_{i_1,\ell-1}, r_{i_1}) \in \Lambda_2$.

Next, we see that aux-$\mathcal{S}$ can compute a WIPOK proof in Step 2a as long as $w_s \neq \bot$. In other words, we see that if aux-$\mathcal{S}$ updates w_s to $(i_1, i_2, \ell - 1, \mathsf{UA}_2, \mathsf{UA}_4, R_2)$ in Step 2b, it is indeed a valid WIPOK witness (that is, $D_{i_2,\ell-1}$ is a commitment to UA_2 and $(h_s, \mathsf{UA}_2, \omega_{i_2}, \mathsf{UA}_4)$ is an accepting UA proof for $(h_s, \tau_s, C_{i_1,\ell-1}, r_{i_1}) \in \Lambda_2$).

1. First, UA_2 is the hash value of a PCP proof π_s that is computed at the end of a previous recursive execution, and from what is observed above, π_s is a valid PCP proof for statement $(h_s, \tau_s, C_{i_1,\ell-1}, r_{i_1}) \in \Lambda_2$. Hence, $(h_s, \mathsf{UA}_2, \omega_{i_2}, \mathsf{UA}_4)$ is an accepting proof for $(h_s, \tau_s, C_{i_1,\ell-1}, r_{i_1}) \in \Lambda_2$

2. Next, since $\mathsf{ua}_{s,\ell-1} \in \mathsf{T}_{\mathsf{UA}}$ is not updated during level-$(\ell-1)$ recursive executions, $D_{i_2,\ell-1}$ is a commitment to UA_2.

Thus, $(i_1, i_2, \ell-1, \mathsf{UA}_2, \mathsf{UA}_4, R_2)$ is a valid WIPOK witness.

Finally, we see that we have $w_s \neq \bot$ when $\mathsf{aux}\text{-}\mathcal{S}$ computes a WIPOK proof in Step 2a. This follows from the following claim.

Claim 5.1. *During the execution of $\mathcal{S}$, any execution of $\mathsf{aux}\text{-}\mathcal{S}$ does not output stuck.*

Proof. We first introduce notations. Recall that an execution of $\mathcal{S}$ involves recursive executions of $\mathsf{aux}\text{-}\mathcal{S}$. We use *block* to denote each execution of $\mathsf{aux}\text{-}\mathcal{S}$. Notice that each block can be identified by the value of ℓ and $\kappa = |\mathsf{trans}|$. A block is in *level* ℓ if the corresponding $\mathsf{aux}\text{-}\mathcal{S}$ is executed with input ℓ. The *child-blocks* of a block are the blocks that are recursively executed by this block; thus, each block has q child-blocks. A block *contains* a slot of a session if the execution of $\mathsf{aux}\text{-}\mathcal{S}$ that corresponds to this block outputs a transcript that includes this slot (i.e., includes both the prover and the verifier message of this slot), where we use *slots* to refer to both Π-slots and UA-slots. A block is *good* w.r.t. a session if this block contains at least two slots of this session but contain neither the first verifier message of this session nor the first prover message of WIPOK of this session.[39]

Given these notations, we prove the claim as follows. From the constructions of $\mathcal{S}$ and $\mathsf{aux}\text{-}\mathcal{S}$, none of $\mathsf{aux}\text{-}\mathcal{S}$ outputs stuck if for every session that reaches Stage 3, there exists a block such that two of its child-blocks are good. (Indeed, if there exists such a block for a session, the first good child-block contains a Π-slot of that session and the second one contains a UA-slot of that session, so a WIPOK witness for that session is computed at the end of the second good child-block.) Thus, it suffices to show that if a session reaches Stage 3, there exists a block such that at least two of its child-blocks are good w.r.t. that session. To show this, it suffices to show that if a session reaches Stage 3, there exists a block such that at least four of its child-blocks contain two or more slots of that session. (Indeed, if four child-blocks contain two or more slots, two of them are good since at most one child-block contains the first verifier message and at most one child-block contains the first prover message of WIPOK.) Assume for contradiction that there exists a session s^* such that s^* reaches Stage 3 but every block has at most three child-blocks that contain two or more slots of session s^*. For $\ell \in \{0, \ldots, d\}$ and $\kappa \in [q^d]$, let $\Gamma_\kappa(\ell)$ be the number of the slots that belong to session s^* and are contained by the block that is identified by ℓ and κ, and let $\Gamma(\ell) \stackrel{\text{def}}{=} \max_\kappa(\Gamma_\kappa(\ell))$. Then, since for each block b,

- at most three child-blocks of b contain two or more slots of s^*, and the other child-blocks contain at most a single slot of s^*, and
- s^* has at most $q-1$ slots that are contained by block b but are not contained by its child-blocks,

[39] The definition of good blocks here is slightly different from that in the technical overview in Section 5.2.2.

we have

$$\Gamma(\ell) \leq 3 \cdot \Gamma(\ell - 1) + (q - 2) \cdot 1 + q - 1 = 3\Gamma(\ell - 1) + 2q - 3 \ .$$

Thus, we have

$$\begin{aligned}
\Gamma(d) &\leq 3\Gamma(d - 1) + 2q - 3 \\
&\leq 3^2\Gamma(d - 2) + 3(2q - 3) + 2q - 3 \\
&\leq \cdots \leq 3^d\Gamma(0) + \sum_{i=0}^{d-1} 3^i(2q - 3) \\
&= 3^d\Gamma(0) + \frac{1}{2}(3^d - 1)(2q - 3) \ .
\end{aligned}$$

From $d = O(1)$ and $\Gamma(0) = 0$, we have $\Gamma(d) = O(q)$. Then, since $\mathcal{S}$ outputs the view of V^* that is generated by the level-d block, there are at most $O(q) = O(n^{\epsilon/2})$ slots of s^* in the simulated transcript. Then, since we have $N_{\mathrm{slot}} = O(n^{\epsilon})$, this contradicts to the assumption that s^* reaches Stage 3. □

From the above three observations, we conclude that $\mathsf{aux}\text{-}\mathcal{S}$ can indeed compute a valid PCP proof π_s and a WIPOK valid witness w_s in Step 2b.

5.4.1.3 Running Time of $\mathcal{S}$

Lemma 5.3. *$\mathcal{S}(x, z)$ runs in polynomial time.*

Proof. We bound the running time of $\mathcal{S}$ as follows. Recall that an execution of $\mathcal{S}$ involves recursive executions of $\mathsf{aux}\text{-}\mathcal{S}$. We identify each execution of $\mathsf{aux}\text{-}\mathcal{S}$ by the value of ℓ and $\kappa = |\mathsf{trans}|$. In the following, we use $\mathsf{aux}\text{-}\mathcal{S}_{\ell,\kappa}$ to denote the execution of $\mathsf{aux}\text{-}\mathcal{S}$ with ℓ and κ. Let $t_{\ell,\kappa}$ be the running time of $\mathsf{aux}\text{-}\mathcal{S}_{\ell,\kappa}$, and let $t_\ell \stackrel{\mathrm{def}}{=} \max_\kappa(t_{\ell,\kappa})$. Then, observe that in the execution of $\mathsf{aux}\text{-}\mathcal{S}_{\ell,\kappa}$, every computation is performed in fixed polynomial time in n except for the following computations.

1. The recursive executions of $\mathsf{aux}\text{-}\mathcal{S}$ (i.e., the executions of $\mathsf{aux}\text{-}\mathcal{S}_{\ell-1,\kappa}, \mathsf{aux}\text{-}\mathcal{S}_{\ell-1,\kappa+q^{\ell-1}}, \ldots$).
2. The generations of the offline proofs (i.e., PCP proofs and their hash values) and the fourth-round UA messages.

Each recursive execution takes at most $t_{\ell-1}$ steps. Furthermore, from the relatively efficient oracle construction property of PCP systems, each offline proof can be generated in $\mathsf{poly}(t_{\ell-1})$ steps. Then, since for each $k \in [q]$ there are a single recursive execution and at most m computations of offline proofs and fourth-round UA proofs, we have

$$t_\ell \leq q \cdot (t_{\ell-1} + m \cdot \mathsf{poly}(t_{\ell-1}) + \mathsf{poly}(n)) + \mathsf{poly}(n) \leq \mathsf{poly}(t_{\ell-1})$$

for any $\ell \in [d]$. Then, since we have $d = O(1)$ and $t_0 = \mathsf{poly}(n)$, we have $t_d = \mathsf{poly}(n)$. Thus, $\mathcal{S}$ runs in polynomial time. □

5.4.1.4 Indistinguishability of Views

Lemma 5.4. *The output of $\mathcal{S}(x,z)$ is computationally indistinguishable from the view of V^*.*

Proof. We prove this lemma by considering a sequence of hybrid experiments, $H_0,\ldots,H_{q^d+1}$. Hybrid H_0 is identical with the real interaction between V^* and honest provers, and H_{q^d+1} is identical with the execution of $\mathcal{S}$. Hybrid H_i ($i\in[q^d]$) is identical with H_{q^d+1} except that, roughly speaking, the simulation stops after simulating the i-th message, and later on the prover messages are generated honestly as in H_0. Formally, we define H_i ($i\in[q^d]$) by using the following hybrid auxiliary simulator aux-$\tilde{\mathcal{S}}_i$, which differs from aux-$\mathcal{S}$ in that it simulates the transcript only until the i-th message and that it simulates the i-th message using true randomness ρ (rather than pseudorandomness ρ_i derived by f-PRG). Though aux-$\tilde{\mathcal{S}}_i$ is very similar to aux-$\mathcal{S}$, we give a complete description of aux-$\tilde{\mathcal{S}}_i$ below. The differences from aux-$\mathcal{S}$ are highlighted by blue color and underline.

Hybrid Auxiliary Simulator aux-$\tilde{\mathcal{S}}_i^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x,z,\ell,\Pi,\mathsf{trans},\sigma,\underline{\tilde{\Pi}_i,\rho})$**:**

Step 1.
`/* Preparing randomness for simulating the next` q^ℓ `messages */`
Let $\kappa := |\mathsf{trans}|$ (i.e., κ be the number of the messages that are included in trans). Then, compute

$$\underline{(\sigma_{i-1},\ldots,\sigma_1,\rho_{i-1},\ldots,\rho_1) := \mathsf{f\text{-}PRG}(\sigma,i-1)}$$

and let $\rho_i := \rho$.

Step 2a: Simulation (base case). If $\ell=0$, do the following.

1. If the next-scheduled message msg is a verifier message, feed trans to $V^*(x,z)$ and receive msg from V^*.

   If the next-scheduled message msg is a prover message of the s-th session ($s\in[m]$), do the following with randomness $\rho_{\kappa+1}$. (If necessary, $\rho_{\kappa+1}$ is expanded by a pseudorandom generator.) Let τ_s be the first-round message of Com of the s-th session, which can be found from trans.

   - If msg is the prover message in a Π-slot, compute $\mathsf{msg}=(C_1,\ldots,C_{N_{\mathrm{col}}})$ by $C_j\leftarrow\mathsf{Com}_{\tau_s}(v_{s,j})$ for every $j\in[N_{\mathrm{col}}]$, where $v_{s,j}$ is read from T_Π.
   - If msg is the prover message in a UA-slot, compute $\mathsf{msg}=(D_1,\ldots,D_{N_{\mathrm{col}}})$ by $D_j\leftarrow\mathsf{Com}_{\tau_s}(\mathsf{ua}_{s,j})$ for every $j\in[N_{\mathrm{col}}]$, where $\mathsf{ua}_{s,j}$ is read from T_{UA}.
   - If msg is the first prover message of WIPOK, compute msg by using w_s as a witness, where w_s is read from T_{W}; if w_s is not a valid witness, abort with output stuck.

- If msg is the second prover message of WIPOK, reconstruct the prover state of WIPOK from w_s and $\rho_1, \ldots, \rho_\kappa$ and then honestly compute msg by using this prover state, where w_s is read from T_{W}.[40]

2. Output msg.

Step 2b: Simulation (recursive case). If $\ell > 0$, do the following.

1. `/* Storing its own code in` T_Π `*/`
   Let T'_Π, $\mathsf{T}'_{\mathsf{UA}}$, T'_{W} be the tables that are obtained by copying the current entries of T_Π, T_{UA}, T_{W}. Let

   $$\Pi_{\text{myself}}(\cdot) \stackrel{\text{def}}{=} \Pi^{\mathsf{T}'_\Pi, \mathsf{T}'_{\mathsf{UA}}, \mathsf{T}'_{\mathsf{W}}}(x, z, \ell, \Pi, \mathsf{trans}, \cdot) \ .$$

   Then, for each $s \in [m]$, if the s-th session has already started in trans, do the following: Let h_s be the hash function chosen by V^* in the s-th session in trans; then, update T_Π by setting $v_{s,\ell} := h_s(\Pi_{\text{myself}})$.

2. Set temporary variables $\mathsf{ctr}_s := 0$ and $\mathsf{tmp}_s := \bot$ for every $s \in [m]$, and a temporary variable new-trans $:= \varepsilon$. Define a function head$(\cdot)$ as $\mathsf{head}(k) \stackrel{\text{def}}{=} \kappa + 1 + (k-1)q^{\ell-1}$.

3. For each $k \in [q]$ such that $\mathsf{head}(k) + q^{\ell-1} < i$, do the following:

   (a) `/* Storing the machine that executes the` k`-th child-block. */`
   Let $\mathsf{T}^{(k)}_\Pi$, $\mathsf{T}^{(k)}_{\mathsf{UA}}$, $\mathsf{T}^{(k)}_{\mathsf{W}}$ be the tables that are obtained by copying the current entries of T_Π, T_{UA}, T_{W}. Then, let

   $$\Pi_k(\cdot) \stackrel{\text{def}}{=} \Pi^{\mathsf{T}^{(k)}_\Pi, \mathsf{T}^{(k)}_{\mathsf{UA}}, \mathsf{T}^{(k)}_{\mathsf{W}}}(x, z, \ell - 1, \Pi, \mathsf{trans} \| \mathsf{new\text{-}trans}, \cdot) \ .$$

   (b) `/* Executing the` k`-th child-block. */`
   Compute

   $$\mathsf{trans}_k := \Pi^{\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}}(x, z, \ell - 1, \Pi, \mathsf{trans} \| \mathsf{new\text{-}trans}, \sigma_{\mathsf{head}(k+1)})$$

   while reading and updating T_Π, T_{UA}, T_{W} for Π.

   (c) For each $s \in [m]$, if the s-th session has already started in trans$\|$new-trans, do the following. Let (h_s, τ_s) be the first-round message of the s-th session.

   Case 1. $\mathsf{ctr}_s = 0$, and trans_k contains a Π-slot of the s-th session.
   `/* Computing offline proof */`
   Let sl denote the Π-slot that is contained by trans_k. (If there are more than one such Π-slot, sl denote the first such one.) Let i_1 denote the *slot-index* of sl, i.e., $i_1 \in [N_{\text{slot}}]$ such that sl is the i_1-th Π-slot in the s-th session. Let $(\overline{C}_{i_1}, r_{i_1})$ denote the messages in sl, where $\overline{C}_{i_1} = (C_{i_1,1}, \ldots, C_{i_1,N_{\text{col}}})$.

[40] From the construction of $\mathcal{S}$ and aux-$\mathcal{S}$, the first prover message of WIPOK in trans must have been computed with witness w_s and randomness in $\rho_1, \ldots, \rho_\kappa$.

i. From $\rho_{\mathsf{head}(k)}, \ldots, \rho_{\mathsf{head}(k+1)-1}$ (which are computed in Step 1), find the randomness R_1 that was used for generating $C_{i_1,\ell-1}$.[41] Then, compute a PCP proof π_s for statement $(h_s, \tau_s, C_{i_1,\ell-1}, r_{i_1}) \in \Lambda_2$ using $(\Pi_k, R_1, \sigma_{\mathsf{head}(k+1)})$ as a witness (see Section 5.4.1.2 for details). Then, compute $\mathsf{UA}_2 := h_s(\pi_s)$.

ii. Update T_{UA} by setting $\mathsf{ua}_{s,\ell-1} := \mathsf{UA}_2$.

iii. Update $\mathsf{tmp}_s := (i_1, \pi_s, \mathsf{UA}_2)$ and $\mathsf{ctr}_s := \mathsf{ctr}_s + 1$.

Case 2. $\mathsf{ctr}_s = 1$, and trans_k contains a UA-slot of the s-th session.

`/* Computing WIPOK witness */`

Let sl denote the UA-slot that is contained by trans_k. (If there are more than one such UA-slot, sl denote the first such one.) Let i_2 be the slot-index of sl. Let $(\overline{D}_{i_2}, \omega_{i_2})$ denote the messages in sl, where $\overline{D}_{i_2} = (D_{i_2,1}, \ldots, D_{i_2,N_{\mathrm{col}}})$.

i. Parse $(i_1, \pi_s, \mathsf{UA}_2) := \mathsf{tmp}_s$. Then, compute a fourth-round UA message UA_4 from the offline proof $(h_s, \pi_s, \mathsf{UA}_2)$ and the third-round UA message ω_{i_2}.

ii. From $\rho_{\mathsf{head}(k)}, \ldots, \rho_{\mathsf{head}(k+1)-1}$ (which are computed in Step 1), find the randomness R_2 that was used for generating $D_{i_2,\ell-1}$. Then, if $w_s = \bot$, update T_{W} by setting $w_s := (i_1, i_2, \ell - 1, \mathsf{UA}_2, \mathsf{UA}_4, R_2)$.

iii. Update $\mathsf{ctr}_s := \mathsf{ctr}_s + 1$.

Case 3. All the other cases.

Do nothing.

(d) Update $\mathsf{new\text{-}trans} := \mathsf{new\text{-}trans} \| \mathsf{trans}_k$.

4. For $k \in [q]$ such that $\mathsf{head}(k) \le i < \mathsf{head}(k+1)$, do the following.

(a) Compute

$$\mathsf{trans}_k := \tilde{\Pi}_i^{\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}}(x, z, \ell - 1, \Pi, \mathsf{trans} \| \mathsf{new\text{-}trans}, \sigma_{\mathsf{head}(k+1)}, \tilde{\Pi}_i, \rho)$$

while reading and updating $\mathsf{T}_\Pi, \mathsf{T}_{\mathsf{UA}}, \mathsf{T}_{\mathsf{W}}$ for $\tilde{\Pi}_i$.

(b) Update $\mathsf{new\text{-}trans} := \mathsf{new\text{-}trans} \| \mathsf{trans}_k$.

5. Output $\mathsf{new\text{-}trans}$.

Now, we formally define the hybrids as follows.

Hybrids $H_0, \ldots, H_{q^d+1}$:

Hybrid H_0 is the same as the real interaction between V^* and honest provers.

Hybrid H_i ($i \in [q^d]$) is the same as the real execution of $\mathcal{S}$ except for the following.

[41] From the construction of $\mathsf{aux}\text{-}\mathcal{S}$, every message in the k-th child-block is computed by using randomness in $\rho_{\mathsf{head}(k)}, \ldots, \rho_{\mathsf{head}(k+1)-1}$.

1. $\mathcal{S}$ obtains trans by executing

$$\text{aux-}\tilde{\mathcal{S}}_i^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x, z, d, \text{aux-}\mathcal{S}, \varepsilon, \sigma_i, \text{aux-}\tilde{\mathcal{S}}_i, \rho_i)$$

rather than

$$\text{aux-}\mathcal{S}^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x, z, d, \text{aux-}\mathcal{S}, \varepsilon, \sigma_{q^d+1}) \ ,$$

where $(\sigma_i, \rho_i) := \text{f-PRG}(\sigma_{i+1}, 1)$ for a randomly chosen seed σ_{i+1} of f-PRG. We remark that in trans, the view of V^* is simulated up until the i-th message (inclusive) in a way that the i-th message is simulated by using ρ_i as randomness.

2. After trans, the simulation of V^*'s view is continued as follows:
   - Every message is computed with true randomness.
   - Every message in Π-slots and UA-slot is generated by committing to 0^n.
   - Every WIPOK that starts after trans is executed with a witness for $x \in L$.
   - Every WIPOK that already started in trans is executed as in aux-$\mathcal{S}$ (i.e., by reconstructing the prover state).

Hybrid H_{q^d+1} is the same as the real execution of $\mathcal{S}$.

From a hybrid argument, it suffices to show the indistinguishability between the outputs of each neighboring hybrids. In the following, we show the indistinguishability in the reverse order, i.e., we show that the output of H_i is indistinguishable from that of H_{i-1} for every $i \in [q^d + 1]$.

Claim 5.2. *The output of H_{q^d+1} and that of H_{q^d} are identically distributed.*

Proof. This claim can be proven by inspection. Notice that the only difference between H_{q^d+1} and H_{q^d} is that in H_{q^d+1}, trans is obtained by executing

$$\text{aux-}\mathcal{S}^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x, z, d, \text{aux-}\mathcal{S}, \varepsilon, \sigma_{q^d+1})$$

whereas in H_{q^d}, trans is obtained by executing

$$\text{aux-}\tilde{\mathcal{S}}_{q^d}^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x, z, d, \text{aux-}\mathcal{S}, \varepsilon, \sigma_{q^d}, \text{aux-}\tilde{\mathcal{S}}, \rho_{q^d})$$

such that $(\sigma_{q^d}, \rho_{q^d}) := \text{f-PRG}(\sigma_{q^d+1}, 1)$. The former execution simulates the q^d messages in trans using the randomness $\rho_1, \ldots, \rho_{q^d}$ that are obtained by $\text{f-PRG}(\sigma_{q^d+1}, q^d)$, whereas the latter execution simulates the first $q^d - 1$ messages using the randomness $\rho_1, \ldots, \rho_{q^d-1}$ that are obtained by $\text{f-PRG}(\sigma_{q^d}, q^d - 1)$ and then simulates the q^d-th message using the randomness ρ_{q^d}. Then, since $(\sigma_{q^d}, \rho_{q^d}) = \text{f-PRG}(\sigma_{q^d+1}, 1)$ in H_{q^d}, it follows from the consistency of f-PRG that the messages in the latter execution are simulated using the randomness $\rho_1, \ldots, \rho_{q^d}$ that are obtained by $\text{f-PRG}(\sigma_{q^d+1}, q^d)$. Hence, the messages in the latter execution are generated identically with those in the former execution. □

Claim 5.3. *The output of H_i and that of H_{i-1} are computationally indistinguishable for every $i \in [q^d]$.*

Proof. To prove this claim, we consider a sequence of intermediate hybrids in which H_i is gradually changed to H_{i-1} as follows.

Hybrid $H_{i:1}$ is the same as H_i except that $\mathcal{S}$ obtains trans by executing

$$\mathsf{aux}\text{-}\tilde{\mathcal{S}}_i^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x, z, d, \mathsf{aux}\text{-}\mathcal{S}, \varepsilon, \sigma_i, \mathsf{aux}\text{-}\tilde{\mathcal{S}}_i, \rho_i)$$

for random σ_i and ρ_i rather than for $(\sigma_i, \rho_i) := \mathsf{f}\text{-}\mathsf{PRG}(\sigma_{i+1}, 1)$.

Notice that in $H_{i:1}$, the i-th message is simulated with true randomness rather than pseudorandomness.

Hybrid $H_{i:2}$ is the same as $H_{i:1}$ except that if the i-th message is a prover message of Com (either in a Π-slot or in a UA-slot), then the message is computed as in the honest prover (i.e., $C_j \leftarrow \mathsf{Com}(0^n)$ or $D_j \leftarrow \mathsf{Com}(0^n)$ for every $j \in [N_{\mathrm{col}}]$).

Hybrid $H_{i:3}$ is the same as $H_{i:2}$ except that if the i-th message is the first prover message of WIPOK, then subsequently all messages in this WIPOK are computed by using a witness for $x \in L$.

From a hybrid argument, it suffices to show the indistinguishability between each neighboring intermediate hybrids.

Claim 5.4. *For every $i \in [q^d]$, the output of $H_{i:1}$ is computationally indistinguishable from that of H_i.*

Proof. The indistinguishability follows immediately from the forward security of $\mathsf{f}\text{-}\mathsf{PRG}$. Assume for contradiction that the output of H_i and that of $H_{i:1}$ are distinguishable. Then, consider the following adversary $\mathcal{D}$ against the forward security of $\mathsf{f}\text{-}\mathsf{PRG}$. On input (σ'_i, ρ'_i), adversary $\mathcal{D}$ internally invokes V^* and simulates H_i for V^* honestly except for the following.

- Rather than executing

$$\mathsf{aux}\text{-}\tilde{\mathcal{S}}_i^{\mathsf{T}_\Pi,\mathsf{T}_{\mathsf{UA}},\mathsf{T}_{\mathsf{W}}}(x, z, d, \mathsf{aux}\text{-}\mathcal{S}, \varepsilon, \sigma_i, \mathsf{aux}\text{-}\tilde{\mathcal{S}}_i, \rho_i)$$

for $(\sigma_i, \rho_i) := \mathsf{f}\text{-}\mathsf{PRG}(\sigma_{i+1}, 1)$, $\mathcal{D}$ executes it for $\sigma_i := \sigma'_i$ and $\rho_i := \rho'_i$.

When σ'_i and ρ'_i are generated by $(\sigma'_i, \rho'_i) := \mathsf{f}\text{-}\mathsf{PRG}(\sigma_{i+1}, 1)$, the output of $\mathcal{D}$ is identically distributed with that of H_i, and when σ'_i and ρ'_i are chosen randomly, the output of $\mathcal{D}$ is identically distributed with that of $H_{i:1}$. Therefore, $\mathcal{D}$ breaks the forward security of $\mathsf{f}\text{-}\mathsf{PRG}$ from the assumption, and thus we reach a contradiction.

□

Claim 5.5. *For every $i \in [q^d]$, the output of $H_{i:2}$ is computationally indistinguishable from that of $H_{i:1}$.*

Proof. It suffices to consider the case that the i-th message msg is a prover message of Com. Note that both in $H_{i:1}$ and $H_{i:2}$, the Com commitments in msg are generated by using true randomness; furthermore, the information about their committed values and randomness are not used in the other messages (e.g., the randomness is not hardwired in the committed machines, and the committed value and the randomness are not used as a witness in the generations of PCP and WIPOK). Thus, the indistinguishability follows from the hiding property of Com. □

Claim 5.6. *For every $i \in [q^d]$, the output of $H_{i:3}$ is computationally indistinguishable from that of $H_{i:2}$.*

Proof. It suffices to consider the case that the i-th message msg is the first prover message of WIPOK. Note that both in $H_{i:2}$ and in $H_{i:3}$, this WIPOK proof is generated with true randomness that is not used anywhere else; furthermore, from Claim 5.1, a valid witness is used both in $H_{i:2}$ and in $H_{i:3}$. Thus, the indistinguishability follows from the witness indistinguishability of WIPOK. □

Claim 5.7. *For every $i \in [q^d]$, the output of H_{i-1} is identically distributed with that of $H_{i:3}$.*

Proof. From the consistency of f-PRG, the first $(i-1)$ messages are computed both in $H_{i:3}$ and in H_{i-1} by using the pseudorandomness that is generated by f-PRG$(\sigma_i, i-1)$ for random σ_i. In addition, the i-th message msg is computed in exactly the same way in $H_{i:3}$ and H_{i-1}. Thus, the claim follows. □

From Claims 5.4, 5.5, 5.6, and 5.7, the output of H_i and that of H_{i-1} are computationally indistinguishable. This completes the proof of Claim 5.3. □

From Claims 5.2 and 5.3, the output of H_0 and that of H_{q^d+1} are indistinguishable. This completes the proof of Lemma 5.4. □

This completes the proof of Lemma 5.1. □

5.4.2 Argument of Knowledge Property

As noted in Remark 5.2, the language Λ_2 in Figure 5.8 is slightly over-simplified, and we can prove the argument-of-knowledge property of cZKAOK only when $\mathcal{H}$ is collision resistant against poly$(n^{\log\log n})$-time adversaries.

Below, we prove the argument-of-knowledge property assuming that $\mathcal{H}$ is collision resistant against poly$(n^{\log\log n})$-time adversaries. By using a trick shown in [BG08], we can extend this proof so that it works even under the assumption that $\mathcal{H}$ is collision resistant against polynomial-time adversaries. The details are given at the end of this section.

Proof of Lemma 5.2. For any cheating prover P^*, let us consider the following extractor E.

- Given oracle access to P^*, the extractor E emulates a verifier of cZKAOK for P^* honestly until the beginning of Stage 3. E then extract a witness from WIPOK using its extractor.

In the following, we assume, as noted above, that $\mathcal{H}$ is collision resistant against $\mathsf{poly}(n^{\log\log n})$-time adversaries.

To show that E outputs a witness for $x \in L$, it suffices to show that the extracted witness is a witness for $(h, \tau, \overline{C}_1, r_1, \overline{D}_1, \omega_1, \ldots, \overline{C}_{N_{\mathrm{slot}}}, r_{N_{\mathrm{slot}}}, \overline{D}_{N_{\mathrm{slot}}}, \omega_{N_{\mathrm{slot}}}) \in \Lambda_1$ only with negligible probability. In the following, we call a witness for $(h, \tau, \overline{C}_1, r_1, \overline{D}_1, \omega_1, \ldots, \overline{C}_{N_{\mathrm{slot}}}, r_{N_{\mathrm{slot}}}, \overline{D}_{N_{\mathrm{slot}}}, \omega_{N_{\mathrm{slot}}}) \in \Lambda_1$ a *fake witness*, and we say that P^* is *bad* if E outputs a fake witness with non-negligible probability. Below, we show that if there exists a bad cheating prover, we can break the collision resistance of $\mathcal{H}$.

We first show the following claim, which roughly states that if there exists a bad P^*, there also exists a prover P^{**} that can prove a statement in Λ_2 with non-negligible probability.

Claim 5.8. *For any ITM P, let us consider an experiment $\mathrm{Exp}_1(n, P)$ in which P interacts with a verifier V as follows.*

1. ***Interactively generating statement.*** *V sends a random $h \in \mathcal{H}_n$ and $\tau \in \{0,1\}^{3n}$ to P. Then, P sends a commitment C of Com to V, and V sends a random $r \in \{0,1\}^{n^2}$ to P.*

2. ***Generating UA proof.*** *P sends a second-round UA message UA_2 of statement $(h, \tau, C, r) \in \Lambda_2$ to V. Then, V sends a third-round UA message ω to P, and P sends a fourth-round UA message UA_4 to V.*

3. *We say that P wins in the experiment if $(h, \mathsf{UA}_2, \omega, \mathsf{UA}_4)$ is an accepting UA proof for $(h, \tau, C, r) \in \Lambda_2$.*

Then, if there exists a bad P^*, there exists a PPT ITM P^{**} that wins in $\mathrm{Exp}_1(n, P^{**})$ with non-negligible probability.

Proof. From the assumption that P^* is bad, for infinitely many n we can extract a fake witness from P^* with probability at least $\delta(n) \stackrel{\mathrm{def}}{=} 1/\mathsf{poly}(n)$. In the following, we fix any such n. From an average argument, there exist $i_1^*, i_2^* \in [N_{\mathrm{slot}}]$ and $j^* \in [N_{\mathrm{col}}]$ such that with probability at least $\delta'(n) \stackrel{\mathrm{def}}{=} \delta(n)/N_{\mathrm{col}}N_{\mathrm{slot}}^2 > \delta(n)/n^3$, we can extract a fake witness $(i_1, i_2, j, \ldots)$ such that $(i_1, i_2, j) = (i_1^*, i_2^*, j^*)$. Then, we consider the following cheating prover P^{**} against Exp_1.

1. P^{**} internally invokes P^* and emulates a verifier of cZKAOK for P^* honestly with the following differences:
   - In Stage 1, P^{**} forwards h and τ from the external V to the internal P^*.
   - In the i_1^*-th Π-slot of Stage 2, P^{**} forwards $C_{i_1^*, j^*}$ from the internal P^* to the external V and forwards r from V to P^*.
   - In Stage 3, P^{**} extracts a witness w from P^* by using the extractor of WIPOK.

2. If w is not a fake witness of the form $(i_1^*, i_2^*, j^*, \ldots)$, P^{**} aborts with output fail. Otherwise, parse $(i_1^*, i_2^*, j^*, \mathsf{UA}_2, \mathsf{UA}_4, R) := w$. Then, P^{**} sends UA_2 to the external V and receives ω.

3. P^{**} rewinds the internal P^* to the point just after P^* sent the Com commitments in the i_2^*-th UA-slot. Then, P^{**} sends ω to P^* as the verifier message of the i_2^*-th UA-slot, interacts with P^* again as an honest verifier, and then extracts a witness w' in Stage 3.

4. If w' is not a fake witness of the form $(i_1^*, i_2^*, j^*, \ldots)$, P^{**} aborts with output fail. Otherwise, parse $(i_1^*, i_2^*, j^*, \mathsf{UA}_2', \mathsf{UA}_4', R') := w'$. Then, P^{**} sends UA_4' to the external V.

To analyze the probability that P^{**} wins in $\mathrm{Exp}_1(n, P^{**})$, we first observe that the transcript of cZKAOK that is internally emulated by P^{**} is "good" with probability at least $\delta'/2$. Formally, let trans be the prefix of a transcript of cZKAOK up until the prover message of the i_2^*-th UA-slot (inclusive). Then, we say that trans is *good* if under the condition that trans is a prefix of the transcript, a fake witness of the form $(i_1^*, i_2^*, j^*, \ldots)$ is extracted from P^* with probability at least $\delta'/2$. From an average argument, the prefix of the transcript is good with probability at least $\delta'/2$ when P^* interacts with an honest verifier of cZKAOK. Then, since a transcript of cZKAOK is perfectly emulated in Step 1 of P^{**}, the prefix of the internally emulated transcript is good with probability at least $\delta'/2$.

We next observe that under the condition that the prefix of the internally emulated transcript is good in Step 1 of P^{**}, P^{**} wins in $\mathrm{Exp}_1(n, P^{**})$ with probability at least $(\delta'/2)^2 - \mathsf{negl}(n)$. First, from the definition of a good prefix, it follows that under the condition that the prefix of the internally emulated transcript is good in Step 1 of P^{**}, the probability that both w and w' are fake witnesses of the form $(i_1^*, i_2^*, j^*, \ldots)$ is at least $(\delta'/2)^2$. Next, if w and w' are fake witnesses, both UA_2 and UA_2' are the committed values of $D_{i_2^*, j^*}$, so $\mathsf{UA}_2 = \mathsf{UA}_2'$ holds except with negligible probability; this means that if w and w' are fake witnesses, $(h, \mathsf{UA}_2, \omega, \mathsf{UA}_4')$ is an accepting UA proof except with negligible probability. Hence, under the aforementioned condition, P^{**} wins in $\mathrm{Exp}_1(n, P^{**})$ with probability at least $(\delta'/2)^2 - \mathsf{negl}(n)$.

By combining the above two observations, we conclude that the probability that P^{**} wins in $\mathrm{Exp}_1(n, P^{**})$ is at least

$$\frac{\delta'}{2}\left(\left(\frac{\delta'}{2}\right)^2 - \mathsf{negl}(n)\right) \geq \frac{1}{\mathsf{poly}(n)} \ .$$

□

Next, we show the following claim, which roughly states that if there exists P^{**} that proves a statement in Λ_2 with non-negligible probability, then we can extract a valid witness from P^{**} with non-negligible probability.

Claim 5.9. *For any ITM E^*, let us consider an experiment $\mathrm{Exp}_2(n, E^*)$ in which E^* interacts with a verifier V as follows.*

1. ***Interactively generating statement.*** *This step is the same as the one in* Exp_1*, where* E^* *plays as* P*. Let* (h, τ, C, r) *be the interactively generated statement.*

2. ***Outputting witness.*** *E outputs $w = (\Pi, R, y)$. We say that E wins in the experiment if w is a valid witness for $(h, \tau, C, r) \in \Lambda_2$.*

Then, if there exists a PPT *ITM* P^{**} *that wins in* $\mathrm{Exp}_1(n, P^{**})$ *with non-negligible probability, there exists a* $\mathsf{poly}(n^{\log\log n})$*-time ITM* E^* *that wins in* $\mathrm{Exp}_2(n, E^*)$ *with non-negligible probability.*

Proof. We first observe that UA satisfies weak/global proof-of-knowledge property even when the statement is generated after the hash function h is chosen, i.e., even when the first-round message of UA is sent before the statement is generated. Roughly speaking, the UA extractor by [BG08] extracts a witness by combining the extractor of the underlying PCP system with an oracle-recovery procedure that (implicitly) recovers a PCP proof for the extractor of the PCP system. A nice property of the UA extractor by [BG08] is that it invokes the oracle-recovery procedure on input a random hash function h that is chosen independently of the statement. Because of this property, the UA extractor can be modified straightforwardly so that it works even when h is chosen before the statement.

We then obtain E^* by simply using the global UA extractor for P^{**}. Since the running time of the global UA extractor is $\mathsf{poly}(n^{\log\log n})$, the running time of E^* is also $\mathsf{poly}(n^{\log\log n})$. □

Finally, we reach a contradiction by showing that given E^* described in Claim 5.9, we can break the collision resistance of $\mathcal{H}$.

Claim 5.10. *If there exists a* $\mathsf{poly}(n^{\log\log n})$*-time ITM* E^* *that wins in* $\mathrm{Exp}_2(n, E^*)$ *with non-negligible probability, there exists a* $\mathsf{poly}(n^{\log\log n})$*-time machine* $\mathcal{A}$ *that breaks the collision resistance of* $\mathcal{H}$*.*

Proof. We consider the following $\mathcal{A}$.

1. Given $h \in \mathcal{H}$, $\mathcal{A}$ internally invokes E^* and emulates $\mathrm{Exp}_2(n, E^*)$ for E^* perfectly except that $\mathcal{A}$ forwards h to E^* in Step 1. Let (h, τ, C, r) and w be the statement and the output of E^* in this emulated experiment.

2. If w is not a valid witness for $(h, \tau, C, r) \in \Lambda_2$, $\mathcal{A}$ aborts with output fail. Otherwise, let $(\Pi, R, y) := w$.

3. $\mathcal{A}$ rewinds E^* to the point just after E^* sent C, and from this point $\mathcal{A}$ emulates $\mathrm{Exp}_2(n, E^*)$ again with fresh randomness. Let (h, τ, C, r') be the statement and w' be the witness in this emulated experiment.

4. If w' is not a valid witness for $(h, C, r') \in \Lambda_2$, $\mathcal{A}$ aborts with output fail. Otherwise, let $(\Pi', R', y') := w'$.

5. $\mathcal{A}$ outputs (Π, Π') if $\Pi \neq \Pi'$ and $h(\Pi) = h(\Pi')$. Otherwise, $\mathcal{A}$ outputs fail.

First, we show that both w and w' are valid witnesses with non-negligible probability. From the assumption that E^* wins in $\text{Exp}_2(n, E^*)$ with non-negligible probability, E^* outputs a valid witness in $\text{Exp}_2(n, E^*)$ with probability $\epsilon \stackrel{\text{def}}{=} 1/\mathsf{poly}(n)$ for infinitely many n. In the following, we fix any such n. Let trans be the prefix of a transcript of $\text{Exp}_2(n, E^*)$ up until E^* sends C (inclusive). We say that trans is *good* if under the condition that trans is a prefix of the transcript, E^* outputs a valid witness with probability at least $\epsilon/2$. From an average argument, the prefix of the internally emulated transcript is good with probability at least $\epsilon/2$. Thus, the probability that both w and w' are valid witnesses is at least $(\epsilon/2)(\epsilon/2)^2 = (\epsilon/2)^3$.

Next, we show that when $\mathcal{A}$ obtains two valid witnesses $w = (\Pi, R, y)$ and $w' = (\Pi', R', y')$, we have $\Pi \neq \Pi'$ and $h(\Pi) = h(\Pi')$ except with negligible probability. First, from the binding property of Com, we have $h(\Pi) = h(\Pi')$ except with negligible probability. (Recall that from the condition that w and w' are valid witnesses, we have $\mathsf{Com}_\tau(h(\Pi); R) = \mathsf{Com}_\tau(h(\Pi'); R') = C$.) Next, since r' is chosen randomly after Π is determined, and since we have

$$\begin{aligned}&\left|\left\{r'' \in \{0,1\}^{n^2} \mid \exists y \in \{0,1\}^n \text{ s.t. } r'' \text{ is a substring of the output of } \Pi(y)\right\}\right| \\ &\leq n^{\log\log n} \cdot 2^n \leq 2^{n+1} \quad ,\end{aligned}$$

the probability that there exists $y'' \in \{0,1\}^n$ such that r' is a substring of the output of $\Pi(y'')$ is at most $2^{n+1}/2^{n^2} = \mathsf{negl}(n)$. Then, since r' is a substring of the output of $\Pi'(y')$, we conclude that we have $\Pi \neq \Pi'$ except with negligible probability.

From the above two observations, we conclude that $\mathcal{A}$ finds a collision of $\mathcal{H}$ with non-negligible probability. □

From Claims 5.8, 5.9, and 5.10, it follows that there exists no bad P^*. Thus, the extractor E outputs a witness for $x \in L$ except with negligible probability. This concludes the proof of Lemma 5.2. □

On the proof of Lemma 5.2 when $\mathcal{H}$ is secure only against poly-time adversaries.

As noted before, we can use a trick by [BG08] to extend the above proof so that it works even when $\mathcal{H}$ is secure only against polynomial-time adversaries. Recall that in the above proof, $\mathcal{H}$ need to be secure against super-polynomial-time adversaries because a collision of $\mathcal{H}$ (i.e., the pair of Π and Π') is found by using the global argument-of-knowledge property of UA. Hence, the overall strategy is to modify the protocol and the proof so that the weak argument-of-knowledge property can be used instead of the global one.

Roughly speaking, the trick by [BG08] works as follows. Recall that, as noted in Section 5.3.2, $\mathcal{H}$ is a hash function family that is obtained by applying Merkle's tree-hashing technique on any length-halving collision-resistant hash function family. From the properties of the tree-hashing, it follows that for any $h \in \mathcal{H}_n$ and $x = (x_1, \ldots, x_{|x|}) \in \{0,1\}^{\leq n^{\log\log n}}$, we can compute short certificates $\mathsf{auth}(x) = \{\mathsf{auth}_i(x)\}_{i \in [|x|]}$ such that given $h(x)$, x_i, and $\mathsf{auth}_i(x)$, one can verify in time polynomial in n that the i-th bit of x is indeed x_i. Furthermore, for any collision (x, x') of $\mathcal{H}$, a collision of the underlying hash function can be found in polynomial time from any pairs of a bit and a certificate

$(x_i, \mathsf{auth}_i(x))$ and $(x'_i, \mathsf{auth}_i(x'))$ such that $x_i \neq x'_i$. Then, the idea of the trick by [BG08] is, instead of finding a collision of $\mathcal{H}$ by extracting the whole of Π and Π', finding a collision of the underlying hash function by extracting Π and Π' in a single bit position along with their certificates. Specifically, in the trick by [BG08], the language Λ_2 is first modified in such a way that a witness includes the certificates of the committed machine so that, if we know a bit position in which Π and Π' differ, we can find a collision of the underlying hash function by extracting Π and Π' in that position along with the corresponding certificates. Then, to make sure that we can find a position in which Π and Π' differ with non-negligible probability, the language Λ_2 is further modified in such a way that the cheating prover is required to commit to the hash value of $\mathsf{ECC}(\Pi)$ instead of the hash value of Π, where ECC is an error-correcting code with constant relative distance and with polynomial-time encoding and decoding; since $\mathsf{ECC}(\Pi)$ and $\mathsf{ECC}(\Pi')$ differ in a constant fraction of their indices, they differ in a randomly chosen position with constant probability. Since we can extract $\mathsf{ECC}(\Pi)$ and $\mathsf{ECC}(\Pi')$ in a single position along with their certificates in time polynomial in n using the weak argument-of-knowledge property of UA, the proof now works under collision resistance against polynomial-time adversaries.

More formally, the trick by [BG08] works as follows. First, we replace the language Λ_2 in Figure 5.8 with the one in Figure 5.9. Next, we modify Claim 5.9 in such a way that E^* is required to extract a witness only implicitly (i.e., output the i-th bit of the witness on input any i); the proof of Claim 5.9 is the same as before except that we use weak argument-of-knowledge property instead of global one. Finally, we modify the proof of Claim 5.10 in such a way that, instead of extracting the whole of w and w', the adversary $\mathcal{A}$ extracts Π and Π' only in a randomly chosen bit position and then extracts the certificates that correspond to that position; also, at the end $\mathcal{A}$ outputs a collision of the underlying hash function if $\mathcal{A}$ can compute it from the extracted bits and certificates. From essentially the same argument as before, it follows that $\mathcal{A}$ finds a collision of the underlying hash function with non-negligible probability. Since the running time of $\mathcal{A}$ is now bounded by a polynomial, we can derive a contradiction even when the underlying hash function is collision resistant only against polynomial-time adversaries.

Language Λ_2:

Let ECC be an error-correcting code with constant relative distance and with polynomial-time encoding and decoding.
$(h, \tau, C, r) \in \Lambda_2$ if and only if there exist

- a machine Π (with some inputs being hardwired) such that $|\Pi| \leq n^{\log\log n}$
- a set of certificates $\{\mathsf{auth}_i\}_{i\in[|\eta|]}$, where $\eta \stackrel{\text{def}}{=} \mathsf{ECC}(\Pi)$
- randomness $R \in \{0,1\}^{\mathsf{poly}(n)}$ for Com
- a string y such that $|y| = n$

such that

- $C = \mathsf{Com}_\tau((|\eta|, h(\eta)); R)$, and
- $\mathsf{auth}_i = \mathsf{auth}_i(\eta)$ for every $i \in [|\eta|]$, and
- $\Pi(y)$ outputs a string that has r as a substring, and $\Pi(y)$ outputs it within $n^{\log\log n}$ steps.

Figure 5.9: A modified version of Λ_2.

Chapter 6

Round-Efficient Black-Box Construction of Composable Multi-Party Computation

In this chapter, we show our last result: A round-efficient black-box construction of composable secure multi-party computation protocols.

6.1 Background

As informally explained in Section 1.1, secure multi-party computation (MPC) protocols enable mutually distrustful parties to compute a functionality without compromising the correctness of the outputs and the privacy of their inputs. In the seminal work of Goldreich et al. [GMW87], it was shown that *general MPC protocols*—MPC protocols that can be used to securely compute any functionality—can be constructed even in the model with malicious adversaries and a dishonest majority.[42]

In this chapter, we consider a *black-box construction* of a general MPC protocol that guarantees *composable security*. Before stating our result, we explain black-box constructions and composable security.

Black-Box Constructions.

A construction of a cryptographic protocol is *black-box* if it uses the underlying cryptographic primitives only in a black-box way (i.e., only through their input/output interfaces). If a construction uses the codes of the underlying primitives, it is *non-black-box*.

As argued by Ishai et al. [IKLP06], constructing black-box constructions is important for both theoretical and practical reasons. Theoretically, it is important because understanding whether non-black-box use of cryptographic primitives is necessary for a cryptographic task is of great interest. Practically, it is important because black-box constructions are typically more efficient than non-black-box ones in terms of both communication and computational complexity. (In fact, most non-black-box constructions

[42] In the following, we consider only such a model.

of general MPC protocols are highly inefficient and hard to implement because they use general NP reductions when executing zero-knowledge proofs.)

Recently, a number of works have studied black-box constructions of general MPC protocols. Ishai et al. [IKLP06] showed the first construction of a general MPC protocol that uses the underlying low-level primitives (such as enhanced trapdoor permutations or homomorphic public-key encryption schemes) in a black-box way. Combined with the subsequent work by Haitner [Hai08], who showed a black-box construction of a (maliciously secure) oblivious transfer protocol based on a semi-honest oblivious transfer protocol, their work gave a black-box construction of a general MPC protocol based on a semi-honest oblivious transfer protocol [HIK+11]. Subsequently, Wee [Wee10] reduced the round complexity to $O(\log^* n)$, and Goyal [Goy11] further reduced the round complexity to $O(1)$.

The security of these black-box protocols are proven in the *stand-alone setting*. Hence, these protocols are secure when a single instance of the protocol is executed at a time.

Composable Security.

A setting that is more general and realistic than the stand-alone setting is the *concurrent setting*, in which many instances of many different protocols are concurrently executed in an arbitrary schedule. A notable difference from the stand-alone setting is that adversaries can now perform a coordinated attack by choosing their messages in an instance based on the executions of the other instances.

As a strong and realistic security notion in the concurrent setting, Canetti [Can01] proposed *universally composable (UC) security*. The main advantage of UC security is *composability*, which guarantees that UC-secure protocols can be composed in such a way that the security of the resultant protocol can be deduced from the security of its components (in other words, UC security enables modular constructions of secure protocols). Composability also guarantees that a protocol remains secure even when it is concurrently executed with any other protocols in any schedule (that is, UC security implies security in the concurrent setting). A UC-secure general MPC protocol was constructed by Canetti et al. [CLOS02] in the *common reference string (CRS) model* (i.e., in a model in which all parties are given a common public string that is chosen by a trusted third party). A black-box construction of a UC-secure general MPC protocol was constructed by Ishai et al. [IPS08] in the $\mathcal{F}_{\text{OT}}$-hybrid model (i.e., in model with the ideal oblivious transfer functionality) and by Choi et al. [CDMW09] in the $\mathcal{F}_{\text{COM}}$-hybrid model (i.e., in the model with the ideal commitment functionality).

UC security, however, turned out to be too strong to achieve in the *plain model*. That is, it was shown that even with non-black-box use of cryptographic primitives, we cannot construct UC-secure general MPC protocols in the model with no trusted setup [CF01, CKL06].

To achieve composable security in the plain model, Prabhakaran and Sahai [PS04] proposed a variant of UC security called *angel-based UC security*. Roughly speaking, angel-based UC security is the same as UC security except that the adversary and the simulator have access to an additional entity—an *angel*—that allows some judicious use of super-polynomial-time resources. Angel-based UC security is weaker

than UC security but guarantees meaningful security in many settings. (For example, angel-based UC security implies *super-polynomial-time simulation (SPS) security* [Pas03, Bar05, GGJS12, PLV12], in which the simulator is allowed to run in super-polynomial time. Hence, angel-based UC security guarantees that whatever an adversary can do in the real world can also be done in the ideal world *in super-polynomial time*.) Furthermore, it was proven that, like UC security, angel-based UC security guarantees composability. (In contrast, SPS security does not guarantee composability.) Prabhakaran and Sahai [PS04] presented a general MPC protocol that satisfies angel-based UC security in the plain model under new assumptions. Subsequently, Malkin et al. [MMY06] constructed another general MPC protocol that satisfies angel-based UC security in the plain model under a new number-theoretic assumption.

Several works have constructed general MPC protocols with angel-based UC security under standard assumptions. Canetti et al. [CLP10, CLP16] constructed a polynomial-round general MPC protocol in angel-based UC security assuming the existence of enhanced trapdoor permutations. Subsequently, Goyal et al. [GLP+15] reduced the round complexity to $\widetilde{O}(\log n)$ under the same assumption. They also showed that by using enhanced trapdoor permutations that are secure against quasi-polynomial-time adversaries, the round complexity of their protocols can be reduced to $O(1)$.

The constructions of these MPC protocols are non-black-box, so they use underlying primitives in a non-black-box way.

Black-Box Constructions of Composable Protocols.

Recently, Lin and Pass [LP12] showed the first black-box construction of a general MPC protocol that guarantees composable security in the plain model. The security of their protocol is proven under angel-based UC security and based on the minimal assumption of the existence of semi-honest oblivious transfer (OT) protocols. The round complexity of their protocol is $O(\max(n^\epsilon, R_{\mathsf{OT}}))$, where $\epsilon > 0$ is an arbitrary constant and R_{OT} is the round complexity of the underlying semi-honest OT protocols. Thus, with enhanced trapdoor permutations (from which we can construct constant-round semi-honest OT protocols), their result gives an $O(n^\epsilon)$-round protocol. Subsequently, a constant-round protocol was constructed by Kiyoshima et al. [KMO14] from constant-round semi-honest OT protocols that are secure against quasi-polynomial-time adversaries and one-way functions that are secure against subexponential-time adversaries.

Summarizing the state-of-the-art, for composable protocols in the plain model, we have

- $\widetilde{O}(\log n)$-round non-black-box constructions under a standard polynomial-time hardness assumption [GLP+15],
- a $O(n^\epsilon)$-round black-box construction under a standard polynomial-time hardness assumption [LP12], and
- $O(1)$-round black-box or non-black-box constructions under standard super-polynomial-time hardness assumptions [GLP+15, KMO14].

Thus, for composable protocols based on standard polynomial-time hardness assumptions, there exists a gap between the round complexity of the non-black-box protocols

($\widetilde{O}(\log n)$ rounds [GLP+15]) and that of the black-box protocols ($O(n^{\epsilon})$ rounds [LP12]). The following is therefore an interesting open question.

> *Does there exist a* ***round-efficient*** *black-box construction of a general MPC protocol that guarantees composability in the plain model under polynomial-time hardness assumptions?*

6.1.1 Our Result

In this chapter, we narrow the gap between the round complexity of black-box composable general MPC protocols and that of non-black-box ones.

Main Theorem. *Assume the existence of R_{OT}-round semi-honest oblivious transfer protocols. Then, there exists a $\max(\widetilde{O}(\log^2 n), O(R_{\mathsf{OT}}))$-round black-box construction of a general MPC protocol that satisfies angel-based UC security in the plain model.*

Recall that, assuming the existence of enhanced trapdoor permutations, we have a constant-round semi-honest OT protocol. Thus, under this assumption, our main theorem gives a $\widetilde{O}(\log^2 n)$-round protocol.

CCA-secure commitment scheme. To prove our main theorem, we construct a $\widetilde{O}(\log^2 n)$-round black-box construction of a *CCA-secure commitment scheme* [CLP10, CLP16, LP12, KMO14, GLP+15] from one-way functions.

Theorem. *Assume the existence of one-way functions. Then, there exists a $\widetilde{O}(\log^2 n)$-round black-box construction of a CCA-secure commitment scheme.*

Roughly speaking, a CCA-secure commitment scheme is a tag-based commitment scheme (i.e., a commitment scheme that takes an n-bit string, a *tag*, as an additional input) such that the hiding property holds even against adversaries that interact with the *committed-value oracle* during the interaction with the challenger. The committed-value oracle interacts with the adversary as an honest receiver in many concurrent sessions of the commit phase. At the end of each session, if the commitment of this session is invalid or has multiple committed values, the oracle returns $\perp$ to the adversary. Otherwise, the oracle returns the unique committed value to the adversary.

Lin and Pass [LP12] showed that in angel-based UC security, an $O(\max(R_{\mathsf{CCA}}, R_{\mathsf{OT}}))$-round general MPC protocol can be obtained in a black-box way from a R_{CCA}-round CCA-secure commitment scheme and a R_{OT}-round semi-honest OT protocol. Thus, we can prove our main theorem by combining the above theorem with the result of Lin and Pass [LP12].

6.1.2 Outline

In Section 6.2, we give an overview of our CCA-secure commitment scheme. In Section 6.3, we give definitions that are used specifically in this chapter. In Section 6.4, we show two building blocks that are used in our CCA-secure commitment scheme. In Section 6.5, we show our CCA-secure commitment scheme and prove its security. In Section 6.6, we show our main theorem.

6.2 Overview of Our CCA-Secure Commitment Scheme

In the previous work on CCA-secure commitment schemes [CLP10, CLP16, LP12, KMO14, GLP+15], *extractability* and *non-malleability* play fundamental roles in the proof of CCA security. Roughly speaking, the CCA security of the existing CCA-secure commitment schemes is proven by reducing it to the hiding property [CLP10, CLP16, LP12] or by showing that the proof of the hiding property goes though even in the presence of the committed-value oracle [KMO14, GLP+15]. During the security proofs, extractability is used to show that the committed-value oracle can be emulated in polynomial time by extracting the committed values from the adversary, and non-malleability is used to show that the emulation of the oracle can be performed without "disturbing" the hiding property [CLP10, CLP16, LP12] or each step of the proof of the hiding property [KMO14, GLP+15].

In this work, we use stronger notions of extractability and non-malleability called *strong extractability* and *one-one CCA security*. In the following, we explain how we construct commitment schemes that satisfy these two notions and how we construct our CCA-secure commitment scheme by using them as building blocks.

6.2.1 Building Block 1: Strongly Extractable Commitment Scheme

A commitment scheme is *strongly extractable* if a rewinding extractor can extract the committed value of a commitment in such a way that the extractor outputs $\perp$ when the commitment is invalid.[43] Strong extractability differs from basic extractability in that it requires the extractor to output $\perp$ when the commitment is invalid; basic extractability, in contrast, allows the extractor to output an arbitrary value when the commitment is invalid (this is called *over-extraction*). A constant-round extractable commitment scheme ExtCom can be constructed in a black-box way from one-way functions [PW09]; however, no black-box construction of a strong extractable commitment scheme has been constructed.

To construct a strongly extractable commitment scheme, we start from the following scheme, in which the cut-and-choose technique is used in the same way as in the previous work on black-box protocols [CDMW08, CDMW09, Wee10, LP12, KMO14].

1. Let v be the value to be committed. Then, the committer computes an $(n+1)$-out-of-$10n$ Shamir's secret sharing $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of value v and commits to each s_j in parallel by using ExtCom.

2. The receiver sends a random subset $\Gamma \subset [10n]$ of size n.

3. For every $j \in \Gamma$, the committer decommits the j-th ExtCom commitment to s_j.

4. The receiver accepts the commitment if and only if the decommitments of ExtCom are valid for every $j \in \Gamma$.

[43] Recall that a commitment is valid if there exists a valid decommitment of this commitment.

For $j \in [10n]$, let us call the j-th ExtCom commitment *the j-th column*. In this scheme, the ExtCom commitments are valid in most columns when the receiver accepts the commitment in Step 4; this is because when the ExtCom commitments are invalid in, say, n columns, at least one of them is chosen by Γ and the receiver rejects the commitment in Step 4 except with exponentially small probability. Since the committed value of a ExtCom commitment can be extracted when it is valid, this implies that the committed shares can be extracted in most columns when the receiver accepts the commitment in Step 4; therefore, when the commitment is valid, the committed value v can be recovered by extracting the committed shares from the ExtCom commitments and then using the error-correcting property of Shamir's secret sharing scheme.[44] Furthermore, by carefully designing the decommit phase as in [CDMW08, CDMW09, Wee10, LP12, KMO14], we can make sure that the extractor outputs $\perp$ when the commitment is invalid.

The problem of this scheme is that we do not know how to prove its hiding property. In particular, since the receiver requests the committer to open adaptively-chosen ExtCom commitments, it can perform *selective opening attacks* [DNRS03], and therefore the hiding property of this scheme cannot be reduced to the hiding property of ExtCom easily.

We therefore modify the scheme and let the receiver commit to Γ at the beginning by using a statistically binding commitment scheme Com. Now, since the receiver no longer chooses the subset Γ adaptively, we can prove the hiding property by using a standard technique. Furthermore, at first sight, the hiding property of Com seems to guarantee that the scheme remains strongly extractable.

In the modified scheme, however, we cannot prove the strong extractability. This is because we can no longer show that most of the ExtCom commitments are valid in an accepting commitment. Consider, for example, that there exists a cheating committer C^* such that after receiving a Com commitment to Γ at the beginning, C^* somehow generates an invalid ExtCom commitment in the j-th column for every $j \notin \Gamma$ and commits to 0^n in the j-th column for every $j \in \Gamma$. Intuitively, it seems that C^* breaks the hiding property of Com. However, we do not know how to use C^* to break the hiding property of Com. To see this, observe the following. Recall that since ExtCom is extractable *with* over-extraction, the extractor of ExtCom may output an arbitrary value when the ExtCom commitment is invalid. Hence, when we extract the committed values of the ExtCom commitments from C^*, the extracted value may be 0^n in every column. Therefore, although C^* behaves differently in ExtCom based on the value of Γ, we do not know how to detect it.

To overcome this problem, we use the commitment scheme wExtCom that was introduced by Goyal et al. [GLOV12]. Roughly speaking, wExtCom is a scheme that is extractable only in a weak sense—extractions may fail with probability at most $1/2$—but is extractable without over-extraction. That is, the extractor may output $\perp$ with probability $1/2$, but when the extractor outputs $v \neq \perp$, the commitment is valid and its committed value is v. Concretely, the commit phase of wExtCom consists of three stages.

[44] Recall that Shamir's secret sharing is also a codeword of Reed-Solomon code.

1. `commit stage`. The committer commits to random $a_0, a_1 \in \{0,1\}^n$ such that $a_0 \oplus a_1 = v$.

2. `challenge stage`. The receiver sends a random bit $ch \in \{0,1\}$.

3. `reply stage`. The committer reveals a_{ch} and decommits the corresponding commitment.

It is easy to see that wExtCom satisfies the following property: For a fixed transcript of the `commit stage`, if a cheating committer returns a valid reply with probability $1/\mathsf{poly}(n)$ for both $ch = 0$ and $ch = 1$, then the committed value can be extracted with probability 1 in expected polynomial time by rewinding the cheating committer.

Using wExtCom, we modify our scheme as follows. After committing to $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ with ExtCom, the committer commits to (s_j, d_j) for each $j \in [10n]$ in parallel by using wExtCom, where (s_j, d_j) is a decommitment of the ExtCom commitment in the j-th column. We then show that most columns are *consistent* in an accepted commitment except with negligible probability, meaning that in most columns on an accepted commitment, the wExtCom commitment is valid and its committed value is a valid decommitment of the corresponding ExtCom commitment except with negligible probability. Toward this end, we observe the following.

- If a cheating committer generates an accepting commitment with non-negligible probability, in wExtCom of more than $9n$ columns the cheating committer returns a valid reply with non-negligible probability for both $ch = 0$ and $ch = 1$. This is because if the cheating committer returns a valid reply with non-negligible probability for both $ch = 0$ and $ch = 1$ in wExtCom of at most $9n$ columns, there are n columns in which the wExtCom commitment is accepted with probability at most $1/2 + \mathsf{negl}(n)$, so the probability that all wExtCom commitments are accepted is negligible.[45]

- Then, from the property of wExtCom, we can extract the committed values of the wExtCom commitments without over-extraction in more than $9n$ columns.

- Then, from the property of the cut-and-choose technique, we can show that in most columns of an accepting commitment, the wExtCom commitment is valid and its committed value is a valid decommitment of the corresponding ExtCom commitment. Note that since the committed values of wExtCom commitments can be extracted without over-extraction, we can show that the cheating committer cannot give invalid wExtCom commitments in many columns.

Then, since the ExtCom commitments are valid in consistent columns, we have that most of the ExtCom commitments are valid whenever the commitment is accepted. We can thus extract the committed value of the scheme without over-extraction as before, i.e., by extracting the committed values of ExtCom commitments and then using the error-correcting property of Shamir's secret sharing scheme.

[45] The formal proof is more complicated because the wExtCom commitments are executed in parallel and thus the columns are not independent of each other.

6.2.2 Building Block 2: One-One CCA-Secure Commitment Scheme

A *one-one CCA-secure commitment scheme*, which is closely related to a *non-malleable commitment scheme*, is one that is CCA secure w.r.t. a restricted class of adversaries that execute only a single session with the committed-value oracle and obtain its committed value from the oracle at the end of the session.[46]

We construct a black-box $O(\log n)$-round one-one CCA-secure commitment scheme by simplifying the CCA-secure commitment scheme of Lin and Pass [LP12] and then applying the "DDN $\log n$ trick" [DDN00, LPV08] on it, where the DDN $\log n$ trick is a transformation by Dolev, Dwork, and Naor (DDN) [DDN00] and has been used to transform a concurrent non-malleable commitment scheme for tags of length $O(\log n)$ to a non-malleable commitment scheme for tags of length $O(n)$ without increasing the round complexity. Roughly speaking, the scheme of [LP12] consists of polynomially-many *rows*—each row is a parallel execution of (a part of) the trapdoor commitment scheme of [PW09]—and a cut-and-choose phase, which forces the committer to give valid and consistent trapdoor commitments in every row. Our idea is to reduce the number of rows from $\mathsf{poly}(n)$ to $\ell(n)$ in the scheme of [LP12], where $\ell(n)$ is the length of the tags. The resultant scheme is no longer CCA secure, but can be shown to be *parallel CCA secure*, i.e., CCA secure w.r.t. a restricted class of adversaries that give only a single parallel queries to the oracle. Then, we set $\ell(n) := O(\log n)$ and apply the DDN $\log n$ trick to the above parallel CCA-secure commitment scheme. It is not hard to show that the resultant scheme is one-one CCA secure.

6.2.3 CCA-Secure Commitment Scheme from the Building Blocks

Now, we explain how we obtain our CCA-secure commitment scheme, CCACom, using a constant-round strongly extractable commitment scheme $\mathsf{sExtCom}$ and a $O(\log n)$-round one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$ as building blocks.

In addition to $\mathsf{sExtCom}$ and $\mathsf{CCACom}^{1:1}$, we use the *concurrently extractable commitment scheme* of Micciancio et al. [MOSV06] in our CCA-secure commitment scheme. Roughly speaking, concurrent extractability guarantees that a rewinding extractor can extract committed values even from polynomially many commitments that are concurrently generated by an adversarial committer. The concurrently extractable commitment scheme of Micciancio et al. [MOSV06], which we denote by CECom, is an abstraction of the preamble stage of the concurrent zero-knowledge protocol of Prabhakaran et al. [PRS02] and is constructed in a black-box way from one-way functions. CECom satisfies even a stronger notion of concurrent extractability called *robust concurrent extractability* [GLP+15], which roughly guarantees that the extractor works even against adversarial committers that additionally participate in an arbitrary external protocol, and furthermore, even though the extractor rewinds the adversarial committers, the external protocol is not rewound during the extraction. CECom satis-

[46] In contrast, a non-malleable commitment scheme is one that is CCA secure w.r.t. a restricted class of adversaries that execute a single session with the oracle and obtain its committed value after completing the session with the oracle *and the session with the challenger*.

fies robust concurrent extractability for a k-round external protocol if a parameter ℓ of CECom (often called "the number of slots" in CECom) satisfies $\ell = \omega(k \log n)$. The round complexity of CECom is $O(\ell)$.

Using sExtCom, $\mathsf{CCACom}^{1:1}$, and CECom as building blocks, we construct CCACom roughly as follows. Let v be the value to be committed to and id be the tag.

1. The receiver commits to a random subset $\Gamma \subset [10n]$ of size n by using $\mathsf{CCACom}^{1:1}$, where the tag of $\mathsf{CCACom}^{1:1}$ is id.

2. The committer computes an $(n+1)$-out-of-$10n$ Shamir's secret sharing $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of value v and commits to each s_j in parallel by using a two-round statistically binding commitment scheme Com. Let $\phi_1, \ldots, \phi_{10n}$ be the commitments and $d_1, \ldots, d_{10n}$ be their decommitments.

3. The committer commits to s_j by using CECom for every $j \in [10n]$ in parallel. Let $\psi_1, \ldots, \psi_{10n}$ be the commitments and $e_1, \ldots, e_{10n}$ be their decommitments. The parameter ℓ of CECom is set as $\ell := O(\log^2 n \log\log n)$ so that we have $\ell = \omega(\log^2 n)$.

4. The committer commits to $u_j \stackrel{\text{def}}{=} (s_j, d_j, e_j)$ by using sExtCom for every $j \in [10n]$ in parallel.

5. The receiver decommits the $\mathsf{CCACom}^{1:1}$ commitment in the first step to Γ.

6. For every $j \in \Gamma$, the committer decommits the j-th sExtCom commitment to $u_j = (s_j, d_j, e_j)$. The receiver verifies whether (s_j, d_j) and (s_j, e_j) are valid decommitments of ϕ_j and $\psi_{\eta,j}$ for every $j \in \Gamma$.

The committed value of a CCACom commitment is defined by the shares that are committed to in the Com commitments (i.e., the committed value is the value that can be reconstructed from these shares).

We prove the CCA security using a hybrid argument. Recall that CCA security requires that the hiding property holds even against adversaries that interact with the committed-value oracle. Toward proving the CCA security of CCACom, we design a series of hybrid experiments in which the CCACom commitment that the adversary receives in the *left session* (the session between the adversary and the challenger) is gradually changed as follows.

- In Hybrid H_0, the CCA-security experiment is executed honestly.

- In Hybrid H_1, the values that are committed to by sExtCom are switched from u_j to $0^{|u_j|}$ for every $j \notin \Gamma$, where Γ is the subset that is committed to by the adversary in the first step.

- In Hybrid H_2, the values that are committed to by CECom are switched from s_j to $0^{|s_j|}$ for every $j \notin \Gamma$.

- In Hybrid H_3, the values that are committed to by Com are switched from s_j to $0^{|s_j|}$ for every $j \notin \Gamma$.

From the security of Shamir's secret sharing, the adversary receives no information about v in H_3. Hence, from a hybrid argument, we can prove CCA security by showing indistinguishability between neighboring hybrid experiments.

Since neighboring hybrids differ only in the values that are committed to in the *row* of sExtCom, CECom, or Com (i.e., the parallel commitments of sExtCom, CECom, or Com), our overall strategy for proving the indistinguishability is to use the hiding property of sExtCom, CECom, and Com. A problem is that the adversary interacts with the committed-value oracle, which extracts the committed values of the *right sessions* (the sessions between the adversary and the committed-value oracle) in super-polynomial time; because of the super-polynomial power of the oracle, the indistinguishability does not follow directly from the hiding property of sExtCom, CECom, and Com. We overcome this problem by showing that the committed-value oracle can be emulated in polynomial time. Specifically, we show that the oracle can be emulated by extracting the committed shares from the rows of CECom using its concurrent extractability and then computing the committed value of each right session from the extracted shares. Roughly speaking, this emulation works because in an accepting right session, the shares committed to in the row of CECom must be "close" to the shares that are committed to in the row of Com (recall that the committed value of a CCACom commitment is defined based on the shares that are committed to in the row of Com); indeed, if they disagree in many locations, the session will be rejected in the last step of the scheme.

In more detail, we prove the indistinguishability between, say, the first and second hybrids in two steps.

Step 1. Prove the indistinguishability assuming that the adversary does not "cheat" in each right session, where, roughly speaking, we say that the adversary cheats in a right session if the adversary commits to $u_j = (s_j, d_j, e_j)$ in the row of sExtCom as specified by the scheme in at most $9n$ locations in an accepting session.

Step 2. Prove that the adversary does not cheat in the right sessions except with negligible probability.

Each step is explained in more detail below.

Step 1. Proving the indistinguishability assuming that the adversary does not cheat. Recall that H_0 and H_1 differ only in the values that are committed to in the row of sExtCom in the left session. For proving indistinguishability between them, we consider new hybrid experiments, G_0 and G_1, such that G_h ($h \in \{0, 1\}$) is the same as H_h except that the committed-value oracle computes the committed value of each right session from the shares that are extracted from the row of CECom (rather than from the row of Com), and those shares are extracted using the robust concurrent extractability of CECom so that the row of sExtCom in the left session is not rewound during the extraction. We then prove the indistinguishability between H_0 and H_1 in two steps.

1. First, we show the indistinguishability between H_h and G_h. Since we assume that the adversary does not cheat in the right sessions, the shares that are committed to in the row of Com and those that are committed to in the row of CECom are 0.9-close. Combined with an error-correcting property of Shamir's secret sharing, their closeness guarantees that the correct committed values of the right seasons are computable even from the shares that are committed to in the row of CECom; hence, the committed-value oracle computes the same value in H_h and G_h, so these two hybrids are indistinguishable.

2. Second, we show the indistinguishability between G_0 and G_1 by using the hiding property of $\mathsf{sExtCom}$. Since these two hybrids run in polynomial time while the adversary is receiving the row of $\mathsf{sExtCom}$ in the left session, and the row of $\mathsf{sExtCom}$ in the left session is not rewound thanks to the robust concurrent extractability of CECom, we can easily design a (non-uniform) reduction from the indistinguishability between G_0 and G_1 to the hiding property of $\mathsf{sExtCom}$.

Combining these two, we obtain the indistinguishability between H_0 and H_1 under the assumption that the adversary does not cheat in the right sessions.

Step 2. Proving that the adversary cannot cheat. Intuitively, the adversary cannot cheat in a right session because the subset that is committed to in the $\mathsf{CCACom}^{1:1}$ commitment of that right session is hidden from the adversary. Indeed, if the subsets are are hidden from the adversary, we can argue that a right session will be rejected in the last step of the scheme when the adversary tries to cheat in that session. However, to formalize this intuition, we need to overcome two obstacles.

Obstacle 1. The adversary interacts with the committed-value oracle, which runs in super-polynomial time. We overcome this obstacle by, again, considering a hybrid in which the oracle is emulated in polynomial time.

Obstacle 2. The challenger cheats in the left session in H_1, H_2, H_3 (recall that in these hybrids, the challenger commits to $0^{|u_j|}$ rather than u_j for every $j \notin \Gamma$ in the row of $\mathsf{sExtCom}$), and thus, the adversary may be able to cheat in a right session by using the messages in the left session. We overcome this obstacle by using the *simulation-soundness* of the cut-and-choose phase. Specifically, since the cheating challenger can be emulated in polynomial time by making a single query to the committed-value oracle of $\mathsf{CCACom}^{1:1}$ (that is, the left session can be emulated in polynomial time if the subset that is committed in to $\mathsf{CCACom}^{1:1}$ is given), the one-one CCA security of $\mathsf{CCACom}^{1:1}$ guarantees that the subset in each right session is hidden even though the challenger cheats in the left session.

More formally, the proof proceeds as follows. Assume for contradiction that the adversary cheats in a right session with non-negligible probability in, say, H_1. Then, there exists a right session such that the adversary cheats with non-negligible probability in this right session but does not cheat except with negligible probability in any right session that completed before this right session; we call this right session the *target right session*. Then, we consider a hybrid experiment that is the same as H_1 except for the following.

- The execution of H_1 is terminated just before the committed-value oracle returns the committed value in the target right session.

- The oracle computes the committed value of each right session from the shares that are extracted from the row of CECom, and those shares are extracted using the robust concurrent extractability of CECom so that the row of $\mathsf{CCACom}^{1:1}$ in the left session, the row of $\mathsf{CCACom}^{1:1}$ in the target right session, and the row of $\mathsf{sExtCom}$ in the target right session are not rewound during the extraction. (Such robust concurrent extraction is possible since the total round complexity of these rows is $O(\log n)$ and the parameter ℓ of CECom satisfies $\ell = \omega(\log^2 n)$.)

Since the oracle returns the committed values only in the right session that terminates before the target right session, and it is assumed that the adversary does not cheat in such right sessions, we can show, as before, that the oracle is correctly emulated in this hybrid. Thus, the adversary cheats in the target right session with non-negligible probability even in this hybrid. Now, since this hybrid runs in polynomial time except when extracting the subset from the $\mathsf{CCACom}^{1:1}$ commitment in the left session, we can break the one-one CCA security of $\mathsf{CCACom}^{1:1}$ in the target right session by extracting the shares committed to in the row of $\mathsf{sExtCom}$ in the target right session and checking the locations where the adversary does not commit to $u_j = (s_j, d_j, e_j)$ in the row of $\mathsf{sExtCom}$ as specified by the scheme, while simulating the left session using the committed-value oracle of $\mathsf{CCACom}^{1:1}$. Hence, we conclude that the adversary does not cheat in the right sessions except with negligible probability.

Remark 6.1. In the above explanation, we assume that $\mathsf{sExtCom}$ has a robust extractability property such that the extraction from the row of $\mathsf{sExtCom}$ is possible even while the $\mathsf{CCACom}^{1:1}$ commitment in the left session is forwarded to the committed-value oracle of $\mathsf{CCACom}^{1:1}$. In the actual proof, we remove the necessity of robust extractability by increasing the number of rows of $\mathsf{sExtCom}$ to $R_{\mathsf{CCA}^{1:1}} + 1$, where $R_{\mathsf{CCA}^{1:1}}$ is the round complexity of $\mathsf{CCACom}^{1:1}$. With $R_{\mathsf{CCA}^{1:1}} + 1$ rows of $\mathsf{sExtCom}$, we can argue that one of the rows of $\mathsf{sExtCom}$ in the target right session does not "interleave" with the $\mathsf{CCACom}^{1:1}$ commitment of the left session, so we extract the values that are committed to in this row of $\mathsf{sExtCom}$. ◇

Remark 6.2. We note that in the above argument, $\mathsf{CCACom}^{1:1}$ need to be one-one CCA secure (rather than just non-malleable) since we need to obtain the committed subset from the oracle immediately after completing the query to the oracle (and possibly before completing the challenge commitment). We also note that $\mathsf{sExtCom}$ must be strongly extractable since otherwise the adversary may give invalid commitments in more than n locations without being detected in the cut-and-choose phase. (As explained in Section 6.2.1, the existence of such an adversary does not contradict the one-one CCA security of $\mathsf{CCACom}^{1:1}$ if over-extraction can occur.) ◇

Combining Steps 1 and 2, we conclude that H_0 and H_1 are indistinguishable. The indistinguishability between other neighboring hybrids can be shown similarly.

Reconstruction procedure $\mathsf{Value}_\Gamma(\boldsymbol{s})$**.** For $\boldsymbol{s} = (s_1, \ldots, s_{10n})$, the output of $\mathsf{Value}_\Gamma(\boldsymbol{s})$ is computed as follows. If $\boldsymbol{s}$ is 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_i = s_i$ for every $i \in \Gamma$, then $\mathsf{Value}(\boldsymbol{s})$ is the value that is decoded from $\boldsymbol{w}$, i.e., $\mathsf{Value}_\Gamma(\boldsymbol{s}) \stackrel{\text{def}}{=} \mathsf{Decode}(\boldsymbol{w})$. Otherwise, $\mathsf{Value}_\Gamma(\boldsymbol{s}) \stackrel{\text{def}}{=} \perp$.

Figure 6.1: Function $\mathsf{Value}_\Gamma(\cdot)$.

6.3 Preliminaries

6.3.1 Shamir's Secret Sharing

We first recall Shamir's secret sharing scheme. (In this thesis, we use only the $(n+1)$-out-of-$10n$ version of it.) To compute a $(n+1)$-out-of-$10n$ secret sharing $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of a value $v \in GF(2^n)$, we choose random $a_1, \ldots, a_n \in GF(2^n)$, let $p(z) \stackrel{\text{def}}{=} v + a_1 z + \cdots + a_n z^n$, and set $s_i := p(i)$ for each $i \in [10n]$. Given $\boldsymbol{s}$, we can recover v by obtaining polynomial $p(\cdot)$ thorough interpolation and then computing $p(0)$. We use $\mathsf{Decode}(\cdot)$ to denote a function that recovers v from $\boldsymbol{s}$ as above.

For any positive real number $x \leq 1$ and any $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ and $\boldsymbol{s'} = (s'_1, \ldots, s'_{10n})$, we say that $\boldsymbol{s}$ and $\boldsymbol{s'}$ are *x-close* if $|\{i \in [10n] \text{ s.t. } s_i = s'_i\}| \geq x \cdot 10n$. If $\boldsymbol{s}$ and $\boldsymbol{s'}$ are not x-close, we say that they are $(1-x)$*-far*. Since the shares generated by $(n+1)$-out-of-$10n$ Shamir's secret sharing scheme are actually a codeword of the Reed-Solomon code with minimum relative distance 0.9, if a (possibly incorrectly generated) sharing $\boldsymbol{s}$ is 0.55-close to a valid codeword $\boldsymbol{w}$, we can recover $\boldsymbol{w}$ from $\boldsymbol{s}$ efficiently by using, for example, the Berlekamp-Welch algorithm.

The following technical lemma will be used in the analyses of our commitment schemes in Sections 6.4.1 and 6.5.

Lemma 6.1. *Let $\boldsymbol{x} = (x_1, \ldots, x_{10n})$ and $\boldsymbol{y} = (y_1, \ldots, y_{10n})$ be any (possibly incorrectly generated) shares of $(n+1)$-out-of-$10n$ Shamir's secret sharing scheme, where some of these shares may be equal to a special error symbol $\perp$. For any set $\Gamma \subset [10n]$ of size n, let $\mathsf{Value}_\Gamma(\cdot)$ be the function that is defined in Figure 6.1. Then, we have $\mathsf{Value}_\Gamma(\boldsymbol{x}) = \mathsf{Value}_\Gamma(\boldsymbol{y})$ if the following three conditions hold.*

1. *For every $i \in [10n]$, if $x_i \neq \perp$, it holds $x_i = y_i$.*
2. *$|\{i \in [10n] \text{ s.t. } x_i = \perp\}| < n \bigwedge \{i \in [10n] \text{ s.t. } x_i = \perp\} \cap \Gamma = \emptyset$.*
3. *$\boldsymbol{x}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_i = x_i$ for every $i \in \Gamma$ or 0.2-far from any such valid codeword.*

Proof. We consider two cases.

Case 1. $\boldsymbol{x}$ is 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_i = x_i$ for every $i \in \Gamma$: First, we observe that $\boldsymbol{y}$ is also 0.9-close to $\boldsymbol{w}$. Since $\boldsymbol{w}$ is a valid codeword, we have $w_i \neq \perp$ for every $i \in [10n]$; thus, we have $x_i \neq \perp$ for every i such that $x_i = w_i$. Also, from the first assumed condition, we have $x_i = y_i$ for every i such that $x_i \neq \perp$. Therefore, we have $y_i = w_i$ for every i such that $x_i = w_i$.

Then, since $\boldsymbol{x}$ is 0.9-close to $\boldsymbol{w}$ from the assumption of this case, we have that $\boldsymbol{y}$ is 0.9-close to $\boldsymbol{w}$.

Next, we observe that $\boldsymbol{w}$ satisfies $w_i = y_i$ for every $i \in \Gamma$. From the second assumed condition, we have $x_i \neq \bot$ for every $i \in \Gamma$. Also, from the first assumed condition, we have $x_i = y_i$ for every i such that $x_i \neq \bot$. Thus, we have $x_i = y_i$ for every $i \in \Gamma$. Then, since we have $w_i = x_i$ for every $i \in \Gamma$ from the assumption of this case, we have $w_i = y_i$ for every $i \in \Gamma$.

Now, since $\boldsymbol{y}$ is 0.9-close to $\boldsymbol{w}$, and $\boldsymbol{w}$ satisfies $w_i = y_i$ for every $i \in \Gamma$, we have $\mathsf{Value}_\Gamma(\boldsymbol{x}) = \mathsf{Value}_\Gamma(\boldsymbol{y}) = \mathsf{Decode}(\boldsymbol{w})$ from the definition of $\mathsf{Value}_\Gamma(\cdot)$.

Case 2. $\boldsymbol{x}$ is 0.2-far from any valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_i = x_i$ for every $i \in \Gamma$: For any valid codeword $\boldsymbol{w'} = (w'_1, \ldots, w'_{10n})$ that satisfies $w'_i = y_i$ for every $i \in \Gamma$, we observe that $\boldsymbol{y}$ is 0.1-far from $\boldsymbol{w'}$. Since we have $x_i \neq \bot$ for every $i \in \Gamma$ (the second assumed condition) and $x_i = y_i$ for every i such that $x_i \neq \bot$ (the first assumed condition), we have $x_i = y_i$ for every $i \in \Gamma$. Then, since we have $w'_i = y_i$ for every $i \in \Gamma$, we have $w'_i = x_i$ for every $i \in \Gamma$. Thus, $\boldsymbol{x}$ is 0.2-far from $\boldsymbol{w'}$ from the assumption of this case. Now, since $\boldsymbol{x}$ and $\boldsymbol{y}$ are 0.9-close from the first and second assumed conditions, it follows that $\boldsymbol{y}$ is 0.1-far from $\boldsymbol{w'}$.

Now, from the definition of $\mathsf{Value}_\Gamma(\cdot)$, we conclude that $\mathsf{Value}_\Gamma(\boldsymbol{x}) = \mathsf{Value}_\Gamma(\boldsymbol{y}) = \bot$.

Notice that from the third assumed condition, either Case 1 or 2 is true. This concludes the proof of Lemma 6.1. □

6.3.2 Strong Computational Binding Property of Commitment Schemes.

We next describe the definition of the strong computational binding property for commitment schemes. (Recall that the standard computational biding property is given in Section 2.3.1.) Roughly speaking, we say that a commitment scheme $\langle C, R\rangle$ satisfies *strong computational binding property* if any PPT committer C^* can generate a commitment that has more than one committed value with at most negligible probability.[47] A formal definition of the strong computational binding property is given below.

Definition 6.1 (Strong computational binding property)**.** *For a commitment scheme $\langle C, R\rangle$ and any PPT adversarial committer C^*, consider the following probabilistic experiment $\mathrm{Exp}^{\mathrm{bind2}}(\langle C, R\rangle, C^*, n, z)$ for any $n \in \mathbb{N}$ and $z \in \{0,1\}^*$.*

> *On input 1^n and auxiliary input z, the adversary C^* interacts with an honest receiver in the commit phase of $\langle C, R\rangle$. Then, C^* is said to win the experiment if there exists two decommitments, (v_0, d_0) and (v_1, d_1), such that*

[47] The standard computational binding property guarantees that for any PPT committer C^*, the commitment that C^* generates cannot be decommitted to more than one value *in polynomial time*. Thus, the commitment that C^* generates is allowed to have more than one committed value.

$v_0 \neq v_1$ but the receiver accepts both (v_0, d_0) and (v_1, d_1) in the decommit phase.

*Then, $\langle C, R \rangle$ is **strongly computationally binding** if for any sequence of auxiliary inputs $\{z_n\}_{n \in \mathbb{N}}$, the probability that C^* wins the experiment $Exp^{\text{bind2}}(\langle C, R \rangle, C^*, n, z_n)$ is negligible.* ◇

6.3.3 Strongly/Weakly Extractable Commitment Schemes

We next describe the definitions of *strongly extractable commitment schemes* and *weakly extractable commitment schemes*, which are variants of the standard extractable commitment schemes (Section 2.3.2).

Strongly Extractable Commitment Schemes. Roughly speaking, an extractable commitment scheme is strongly extractable if no over-extraction occurs during the extraction. (Recall that, as explained in Section 2.3.2, we say that over-extraction occurs during the extraction if the extractor extracts an arbitrary value (rather than $\perp$) from an invalid commitment.) Formally, a statistically binding commitment scheme $\langle C, R \rangle$ is *strongly extractable* if there exists an expected polynomial-time extractor E such that for any PPT committer C^*, the extractor E^{C^*} outputs a pair (τ, σ) that satisfies the following properties.

- τ is identically distributed with the view of C^* that interacts with an honest receiver R in the commit phase of $\langle C, R \rangle$. Let c_τ be the commitment that C^* gives in τ.
- If c_τ is invalid, then $\sigma = \perp$ except with negligible probability.
- If c_τ is valid, then it is statistically impossible to decommit c_τ to any value other that σ.

Weakly Extractable Commitment Schemes. Roughly speaking, an extractable commitment scheme is weakly extractable if the extraction can fail with probability $1/2$ but no over-extraction occurs during the extraction. Formally, a commitment scheme $\langle C, R \rangle$ is *weakly extractable* if there exists an expected polynomial-time extractor E such that for any PPT committer C^*, the extractor E^{C^*} outputs a pair (τ, σ) that satisfies the following properties.

- τ is identically distributed with the view of C^* that interacts with an honest receiver R in the commit phase of $\langle C, R \rangle$. Let c_τ be the commitment that C^* gives in τ.
- The probability that c_τ is accepting and $\sigma = \perp$ is at most $1/2$.
- If $\sigma \neq \perp$, then c_τ is valid and it is statistically impossible to decommit c_τ to any value other than σ.

Let Com be any two-round statistically binding commitment scheme that can be constructed from one-way functions in a black-box way.

Commit Phase

The committer C and the receiver R take common input 1^n, and C additionally takes private input $v \in \{0,1\}^n$. To commit to v, the committer C does the following with the receiver R.

`commit stage.` C chooses a pair of random n-bit strings (a_0, a_1) such that $a_0 \oplus a_1 = v$. Then, C commits to a_0 and a_1 by using Com. For each $b \in \{0,1\}$, let c_b be the commitment to a_b.

`challenge stage.` The receiver R sends a random bit $e \in \{0,1\}$ to C.

`reply stage.` C decommits c_e to a_e.

Decommit Phase

C sends v to R and decommits c_0 and c_1 to a_0 and a_1. Then, R checks whether $a_0 \oplus a_1 = v$.

Figure 6.2: Weakly extractable commitment scheme wExtCom [GLOV12].

There exists a four-round weakly extractable commitment scheme wExtCom based on one-way functions [GLOV12], and it uses the underlying one-way function in a black-box way. wExtCom is shown in Figure 6.2. We note that given two accepted transcripts of wExtCom such that `commit stage` is identical but `challenge stage` is different, we can extract the committed value.

6.3.4 Trapdoor Commitment Schemes

We next recall *trapdoor commitment schemes* [PW09]. Roughly speaking, trapdoor commitment schemes are commitment schemes such that there exists a simulator that can generate a simulated commitment and can later decommit it to any value. Pass and Wee [PW09] showed that the black-box scheme TrapCom in Figure 6.3 is a trapdoor bit commitment. TrapCom is not statistically binding, but it satisfies the strong computational binding property. (The strong computational binding property holds since if an adversarial committer C^* generates a TrapCom commitment that can be decommitted to both 0 and 1, we can break the hiding property of Com using C^* by extracting the committed values of the ExtCom commitments from C^* and then computing the committed value e of Com from them.) Pass and Wee also showed that by running TrapCom in parallel, we can obtain a black-box trapdoor commitment scheme PTrapCom for multiple bits. PTrapCom also satisfies the strong computational binding property.

Commit Phase

To commit to $\sigma \in \{0,1\}$ on common input 1^n, the committer C does the following with the receiver R.

Step 1. R chooses a random n-bit string $e = (e_1, \ldots, e_n)$ and commits to e by using Com.

Step 2. For each $i \in [n]$, the committer C chooses a random $\eta_i \in \{0,1\}$ and then sets

$$v_i := \begin{pmatrix} v_i^{00} & v_i^{01} \\ v_i^{10} & v_i^{11} \end{pmatrix} = \begin{pmatrix} \eta_i & \eta_i \\ \sigma \oplus \eta_i & \sigma \oplus \eta_i \end{pmatrix} .$$

Then, for each $i \in [n]$, $\alpha \in \{0,1\}$, and $\beta \in \{0,1\}$ in parallel, C commits to $v_i^{\alpha\beta}$ by using ExtCom; let $(v_i^{\alpha\beta}, d_i^{\alpha\beta})$ be the corresponding decommitment.

Step 3. R decommits the commitment in Step 1 to e.

Step 4. For each $i \in [n]$, C sends $(v_i^{e_i 0}, d_i^{e_i 0})$ and $(v_i^{e_i 1}, d_i^{e_i 1})$ to R. Then, R checks whether these are valid decommitments and whether $v_i^{e_i 0} = v_i^{e_i 1}$.

Decommit Phase

C sends σ and random $\gamma \in \{0,1\}$ to R. In addition, for every $i \in [n]$, C sends $(v_i^{0\gamma}, d_i^{0\gamma})$ and $(v_i^{1\gamma}, d_i^{1\gamma})$ to R. Then, R checks whether $(v_i^{0\gamma}, d_i^{0\gamma})$ and $(v_i^{1\gamma}, d_i^{1\gamma})$ are valid decommitments and whether $v_0^{0\gamma} \oplus v_0^{1\gamma} = \cdots = v_n^{0\gamma} \oplus v_n^{1\gamma} = \sigma$.

Figure 6.3: Black-box trapdoor bit commitment scheme $\mathsf{TrapCom}$.

6.4 Building Blocks

In this section, we construct a constant-round strongly extractable commitment scheme and a $O(\log n)$-round one-one CCA-secure commitment scheme. Both schemes are used in our $\widetilde{O}(\log^2 n)$-round CCA-secure commitment scheme in Section 6.5 as building blocks.

6.4.1 Strongly Extractable Commitment Scheme

Using one-way functions in a black-box way, we construct a constant-round strongly extractable commitment scheme $\mathsf{sExtCom}$. Recall that a commitment scheme is strongly extractable if a rewinding extractor outputs a correct committed value when the commitment is valid and outputs $\perp$ when the commitment is invalid.

Lemma 6.2. *Assume the existence of one-way functions. Then, there exists a constant-round strongly extractable commitment scheme* $\mathsf{sExtCom}$ *that uses the underlying one-way function only in a black-box way.*

Proof. The scheme sExtCom is shown in Figure 6.4, in which we use the following tools (all of which can be constructed from one-way functions in a black-box way).

- A two-round statistically binding commitment scheme Com. (A concrete example is Naor's commitment scheme in Section 2.3.1, which can be constructed from one-way functions in a black-box way [Nao91, HILL99].)
- A constant-round extractable commitment scheme ExtCom. (A concrete example is the extractable commitment scheme of Pass and Wee [PW09] in Section 2.3.2, which can be constructed from one-way functions in a black-box way.)
- The constant-round weakly extractable commitment scheme wExtCom of Goyal et al. [GLOV12]. (See Section 6.3.3.)

We prove the binding property and the hiding property in Section 6.4.1.1 and the strong extractability in Section 6.4.1.2.

6.4.1.1 Proofs of Binding and Hiding

First, we show that sExtCom is statistically binding and computationally hiding. The binding property follows directly from that of ExtCom. To show the hiding property, we consider the following hybrid experiments for any PPT cheating receiver R^* and each $b \in \{0,1\}$.

Hybrid $H_0^b(n,z)$ is an experiment in which R^* takes input 1^n and auxiliary input z and receives a sExtCom commitment to σ_b from an honest committer, where (σ_0,σ_1) is the challenge values that R^* chooses at the beginning. The output of $H_0^b(n,z)$ is that of R^*.

Hybrid $H_1^b(n,z)$ is the same as $H_0^b(n,z)$ except that the sExtCom commitment from the committer is modified as follows.

- In Step 1, the committed value Γ is extracted by brute force.
- In Step 2, the committer commits to $0^{|s_j|}$ instead of s_j for every $j \notin \Gamma$.
- In Step 3, the committer commits to $(0^{|s_j|}, 0^{|d_j|})$ instead of (s_j, d_j) for every $j \notin \Gamma$.

Let $\mathsf{H}_i^b(n,z)$ be the random variable representing the output of $H_i^b(n,z)$ for $i \in \{0,1\}$ and $b \in \{0,1\}$. From the construction, R^* receives no information about b in $H_1^b(n,z)$ for each $b \in \{0,1\}$, so the distributions of $\mathsf{H}_1^0(n,z)$ and $\mathsf{H}_1^1(n,z)$ are identical. Hence, from a hybrid argument, we can show the hiding property by showing that $\mathsf{H}_0^b(n,z)$ and $\mathsf{H}_1^b(n,z)$ are indistinguishable for each $b \in \{0,1\}$. Assume for contradiction that there exists $b \in \{0,1\}$ such that for infinitely many n, there exists $z \in \{0,1\}^*$ such that $\mathsf{H}_0^b(n,z)$ and $\mathsf{H}_1^b(n,z)$ are distinguishable with advantage $1/\mathsf{poly}(n)$. Fix any such b, n, and z. From an average argument, there exists a transcript ρ of Step 1 such that under the condition that the transcript of Step 1 is ρ, $\mathsf{H}_0^b(n,z)$ and $\mathsf{H}_1^b(n,z)$ are distinguishable with advantage $1/\mathsf{poly}(n)$. Let Γ be the subset that is committed to in ρ. Since we can execute $H_1^b(n,z)$ from ρ in polynomial time given ρ and Γ, by using a standard technique we can break the hiding property of either ExtCom or wExtCom by using ρ and Γ as auxiliary input. Thus, we reach a contradiction.

Commit Phase

The committer C and the receiver R take common input 1^n, and C additionally takes private input $\sigma \in \{0,1\}^n$. To commit to σ, the committer C does the following with the receiver R.

Step 1. R commits to a random subset $\Gamma \subset [10n]$ of size n by using Com.

Step 2. C computes an $(n+1)$-out-of-$10n$ Shamir's secret sharing $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of value σ. Then, for each $j \in [10n]$ in parallel, C commits to s_j by using ExtCom. Let $\phi_1, \ldots, \phi_{10n}$ be the commitments and $d_1, \ldots, d_{10n}$ be the decommitments.

Step 3. For each $j \in [10n]$ in parallel, C commits to (s_j, d_j) by using $\mathsf{wExtCom}$. Let $\psi_0, \ldots, \psi_{10n}$ be the commitments.

Step 4. R decommits the commitment in Step 1 to Γ.

Step 5. C decommits the j-th $\mathsf{wExtCom}$ commitment ψ_j in Step 3 to (s_j, d_j) for each $j \in \Gamma$. R checks whether (s_j, d_j) is a valid decommitment of the j-th ExtCom commitment ϕ_j in Step 2 for every $j \in \Gamma$.

Decommit Phase

- C sends σ and decommits the ExtCom commitments $\phi_1, \ldots \phi_{10n}$ in Step 2 to $s_1, \ldots, s_{10n}$.
- R accepts if and only if the following holds w.r.t. $\boldsymbol{s} = (s_1, \ldots, s_{10n})$, where s_j is defined to be $\perp$ if the decommitment of ϕ_j is invalid.
  - $\boldsymbol{s}$ is 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_j = s_j$ for every $j \in \Gamma$, and $\boldsymbol{w}$ is a codeword of σ.

Figure 6.4: Strongly extractable commitment scheme $\mathsf{sExtCom}$.

6.4.1.2 Proof of Strong Extractability.

Next, we show that $\mathsf{sExtCom}$ is strongly extractable. That is, we show that an extractor extracts a correct committed value from a valid $\mathsf{sExtCom}$ commitment and extracts $\perp$ from an invalid one except with negligible probability.

We first remark that from the construction of the decommit phase of $\mathsf{sExtCom}$, the committed value of $\mathsf{sExtCom}$ is defined as follows.

Definition 6.2 (Committed value of $\mathsf{sExtCom}$)**.** *If the shares $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ that are committed to in Step 2 are 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_j = s_j$ for every $j \in \Gamma$, the committed value of a $\mathsf{sExtCom}$ commitment is $\mathsf{Decode}(\boldsymbol{w})$. Otherwise, the committed value is $\perp$ (i.e., the commitment is invalid).* $\diamond$

We notice that the function $\mathsf{Value}_\Gamma(\cdot)$ in Figure 6.1 (Section 6.3.1) computes the com-

mitted value of a sExtCom commitment as above on input the shares $\boldsymbol{s}$ that are committed to in Step 2.

Our extractor E extracts the committed value of a sExtCom commitment by extracting the committed values of the ExtCom commitments in Step 2. Formally, for any PPT cheating committer C^*, the extractor E does the following.

- E internally invokes C^* and interacts with C^* as a receiver honestly except that in Step 2, E extracts the committed values of the ExtCom commitments by using the extractability of ExtCom. Let τ be the view of internal C^*. If the sExtCom commitment in τ is rejecting or E fails to extract the committed values of the ExtCom commitments in Step 2, E sets $\widetilde{\sigma} := \bot$. Otherwise, E sets $\widetilde{\sigma} := \mathsf{Value}_\Gamma(\widetilde{\boldsymbol{s}})$, where $\widetilde{\boldsymbol{s}}$ is the shares that are extracted from the ExtCom commitments and Γ is the subset that is committed to in Step 1. E then outputs $(\tau, \widetilde{\sigma})$.

From the extractability of ExtCom, the simulated view τ is identically distributed with the real view. Hence, it remains to show that $\widetilde{\sigma}$ is a committed value of τ except with negligible probability.

Fix any PPT cheating committer C^*. Without loss of generality, we assume that C^* is deterministic.

First, we show that the extracted value $\widetilde{\sigma}$ is indeed equal to a committed value of the simulated view τ as long as the ExtCom commitments in Step 2 in τ are "good."

Definition 6.3 (Good ExtCom commitments in Step 2)**.** *In a* sExtCom *commitment, we say that the* ExtCom *commitments in Step 2 are* ***good*** *if all of the following conditions hold.*

- *Their committed values* $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ *are uniquely determined.*
  (That is, none of them has more than one committed value.)
- $\left|\left\{j \in [10n] \text{ s.t. } s_j = \bot\right\}\right| < 0.5n$.
  (That is, less than $0.5n$ *of them are invalid.)*
- $\boldsymbol{s}$ *is either* 0.9*-close to a valid codeword* $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ *that satisfies* $w_j = s_j$ *for every* $j \in \Gamma$ *or* 0.2*-far from any such valid codeword.*

$\diamond$

Claim 6.1. *Assume that in the interaction between C^* and an honest receiver, the probability that the* sExtCom *commitment from C^* is accepting but the* ExtCom *commitments in Step 2 are not good is negligible. Then, in the execution of E, the extracted value $\widetilde{\sigma}$ is a correct committed value of the* sExtCom *commitment in τ except with negligible probability.*

Proof. When the sExtCom commitment in τ is rejecting, E sets $\widetilde{\sigma} := \bot$, which is a correct committed value of this sExtCom commitment. Hence, it remains to show that the probability that the sExtCom commitment in τ is accepting but $\widetilde{\sigma}$ is not its committed value is negligible.

Let BAD be the event that in the execution of E, the sExtCom commitment in the simulated view τ is accepting but the extracted value $\widetilde{\sigma} = \mathsf{Value}_\Gamma(\widetilde{s})$ is not a committed value of it. Our goal is to show that BAD occurs only with negligible probability. Since the simulated view τ is identically distributed with the real view of C^*, from our assumption the probability that the sExtCom commitment in τ is accepting but the ExtCom commitments in Step 2 of it are not good is negligible. Hence, it suffices to show that under the condition that those ExtCom commitments are good, BAD occurs only with negligible probability. Furthermore, since the extraction from ExtCom succeeds except with negligible probability, and the values extracted from valid ExtCom commitments are the correct committed values except with negligible probability, it suffices to show that under the conditions that in the sExtCom commitment in τ,

- the ExtCom commitments in Step 2 are good, and
- the (unique) committed value of each valid ExtCom commitment is correctly extracted,

BAD occurs only with negligible probability. Then, we notice that under the above conditions, we have the following when the sExtCom commitment in τ is accepting.

1. For every $j \in [10n]$, if $s_j \neq \perp$, it holds $s_j = \widetilde{s}_j$.

   (This is because of the assumption that the correct committed value is extracted from every valid ExtCom commitment.)

2. $\left|\left\{j \text{ s.t. } s_j = \perp\right\}\right| < 0.5n \wedge \left\{j \text{ s.t. } s_j = \perp\right\} \cap \Gamma = \emptyset$.

   (This is because the sExtCom commitment would be rejected in Step 5 if $\{j \text{ s.t. } s_j = \perp\} \cap \Gamma \neq \emptyset$.)

3. $\boldsymbol{s}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_j = s_j$ for every $j \in \Gamma$ or 0.2-far from any such valid codeword.

Hence, using Lemma 6.1 in Section 6.3.1, we conclude that under the above conditions, we have $\mathsf{Value}_\Gamma(\widetilde{s}) = \mathsf{Value}_\Gamma(s)$ (i.e., $\mathsf{Value}_\Gamma(\widetilde{s})$ is equal to the committed value) when the sExtCom commitment in τ is accepting. Thus, BAD never occurs under the above conditions. This completes the proof of Claim 6.1. □

It remains to show that in the interaction between C^* and an honest receiver, the probability that the sExtCom commitment from C^* is accepting but the ExtCom commitments in Step 2 are not good is negligible. Recall that the ExtCom commitments are good if their committed values $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ are uniquely determined, at least $9.5n$ of them are valid, and $\boldsymbol{s}$ is either 0.9-close to a valid codeword $\boldsymbol{w}$ that satisfies $w_j = s_j$ for every $j \in \Gamma$ or 0.2-far from any such codewords. We show the following two claims.

Claim 6.2. *In the interaction between C^* and an honest receiver, the probability that the* sExtCom *commitment from C^* is accepting but at least* $0.5n$ ExtCom *commitments in Step 2 are invalid is negligible.*

Claim 6.3. *In the interaction with C^* and an honest receiver, the probability that the* sExtCom *commitment from C^* is accepting but either of the following conditions does not hold is negligible.*

- *The committed values $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of the* ExtCom *commitments are uniquely determined.*
- *$\boldsymbol{s}$ is either 0.9-close to a valid codeword $\boldsymbol{w}$ that satisfies $w_j = s_j$ holds for every $j \in \Gamma$ or 0.2-far from any such codewords.*

Proof of Claim 6.2.

In this proof, we use the following notations. For $j \in [10n]$, the *j-th column* is the pair of the j-th ExtCom commitment in Step 2 and the j-th wExtCom commitment in Step 3. A column is *consistent* if the committed value of the wExtCom commitment is a valid decommitment of the ExtCom commitment in that column; otherwise, the column is *inconsistent*. *C^* cheats* if all of the following conditions hold: every wExtCom commitment is accepting, the j-th column is consistent for every $j \in \Gamma$, and at least $0.5n$ columns are inconsistent.

In the following, we show that C^* cheats only with negligible probability. This suffices to prove the claim because from the definition of the cheating, C^* cheats whenever the sExtCom commitment from C^* is accepting but at least $0.5n$ ExtCom commitments in Step 2 are invalid.

Assume for contradiction that there exists a constant c such that C^* cheats with probability at least $1/n^c$ for infinitely many n. Fix any such c and n.

We derive a contradiction by constructing an adversary $\mathcal{B}$ that breaks the hiding property of Com. For random subsets $\Gamma_0, \Gamma_1 \subset [10n]$ of size n, $\mathcal{B}$ tries to distinguish a Com commitment to Γ_0 from a Com commitment to Γ_1 as follows. $\mathcal{B}$ internally invokes C^* and interacts with it as a receiver of sExtCom honestly except for the following.

- In Step 1, $\mathcal{B}$ receives a Com commitment from the external committer (who commits to either Γ_0 or Γ_1) and forwards the commitment to C^* as the commitment in Step 1.
- If Step 3 is accepting (i.e., all of the wExtCom commitments are accepting), $\mathcal{B}$ does the following repeatedly: $\mathcal{B}$ rewinds C^* to the point just before $\mathcal{B}$ sends the challenge bits of the wExtCom commitments to C^*; then, $\mathcal{B}$ sends new random challenge bits to C^* and receives the replies from C^*. $\mathcal{B}$ repeats this rewinding until it obtains other n^{c+3} accepted transcripts of Step 3. If the number of the rewinding exceeds n^{3c+4}, $\mathcal{B}$ terminates and outputs fail. Otherwise, $\mathcal{B}$ outputs 1 if and only if all of the following conditions hold.
  1. From the $n^{c+3} + 1$ accepted transcripts of Step 3 (the first one and the subsequent n^{c+3} ones), $\mathcal{B}$ can extract the committed values of the wExtCom commitments in at least $9.9n$ columns.
  2. In at least $0.4n$ columns of these $9.9n$ columns, the extracted values are not valid decommitments of the ExtCom commitments.

3. For every $j \in \Gamma_1$, either the extraction from the j-th column fails or the value extracted from the j-th column is a valid decommitment of the ExtCom commitment of the j-th column.

In the following, the first transcript that $\mathcal{B}$ generates in Step 3 is called the *main thread* and the other n^{c+3} accepted transcripts are called the *look-ahead threads*.

First, we analyze the adversary $\mathcal{B}'$ that is the same as $\mathcal{B}$ except that $\mathcal{B}'$ does not terminate even after rewinding C^* more than n^{3c+4} times. When $\mathcal{B}'$ receives a commitment to Γ_0, the internal C^* receives no information about Γ_1, so the probability that the extracted values are not valid decommitments of the ExtCom commitments in at least $0.4n$ columns but are valid decommitments in all the columns selected by Γ_1 is exponentially small. Hence, when $\mathcal{B}'$ receives a commitment to Γ_0, $\mathcal{B}'$ outputs 1 only with exponentially small probability. In the following, we show that when $\mathcal{B}'$ receives a commitment to Γ_1, $\mathcal{B}'$ outputs 1 with probability $1/\mathsf{poly}(n)$. Let CHEAT be the event that C^* cheats on the main thread, and $\mathsf{EXTRACT}$ be the event that $\mathcal{B}'$ succeeds in extracting the committed values of the $\mathsf{wExtCom}$ commitments from at least $9.9n$ columns. Since over-extraction never occurs in the extraction from $\mathsf{wExtCom}$, $\mathcal{B}'$ outputs 1 whenever CHEAT and $\mathsf{EXTRACT}$ occur. Hence, to show that $\mathcal{B}'$ outputs 1 with probability at least $1/\mathsf{poly}(n)$, it suffices to show that we have

$$\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT}\right] \geq \frac{1}{\mathsf{poly}(n)} \quad . \tag{6.1}$$

For any prefix ρ of the transcript between C^* and an honest receiver up until the challenge bits of $\mathsf{wExtCom}$ (exclusive), let PREFIX_ρ be the event that ρ is a prefix of the main thread. Since C^* cheats with probability at least $1/n^c$, from an average argument we have $\Pr\left[\mathsf{CHEAT} \mid \mathsf{PREFIX}_\rho\right] \geq 1/2n^c$ with probability at least $1/2n^c$ over the choice of ρ (i.e., over the distribution of ρ in the interaction between C^* and an honest receiver). Let Δ be the set of prefixes with which $\Pr\left[\mathsf{CHEAT} \mid \mathsf{PREFIX}_\rho\right] \geq 1/2n^c$ holds. As noted above, we have $\sum_{\rho\in\Delta} \Pr\left[\mathsf{PREFIX}_\rho\right] \geq 1/2n^c$. Hence, we have

$$\begin{aligned}
\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT}\right] &\geq \sum_{\rho\in\Delta} \Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT} \mid \mathsf{PREFIX}_\rho\right] \cdot \Pr\left[\mathsf{PREFIX}_\rho\right] \\
&\geq \min_{\rho\in\Delta}\left(\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT} \mid \mathsf{PREFIX}_\rho\right]\right) \cdot \sum_{\rho\in\Delta} \Pr\left[\mathsf{PREFIX}_\rho\right] \\
&\geq \frac{1}{2n^c} \min_{\rho\in\Delta}\left(\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT} \mid \mathsf{PREFIX}_\rho\right]\right) \quad . \qquad (6.2)
\end{aligned}$$

Thus, to show Equation (6.1), it suffices to show that for any $\rho \in \Delta$, we have

$$\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT} \mid \mathsf{PREFIX}_\rho\right] \geq \frac{1}{\mathsf{poly}(n)} \quad . \tag{6.3}$$

Fix any $\rho^* \in \Delta$. From the definition of Δ, we have

$$\Pr\left[\mathsf{CHEAT} \mid \mathsf{PREFIX}_{\rho^*}\right] \geq \frac{1}{2n^c} \quad . \tag{6.4}$$

Thus, we have

$$\begin{aligned}
&\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT} \mid \mathsf{PREFIX}_{\rho^*}\right] \\
&= \Pr\left[\mathsf{CHEAT} \mid \mathsf{PREFIX}_{\rho^*}\right] \cdot \Pr\left[\mathsf{EXTRACT} \mid \mathsf{PREFIX}_{\rho^*} \wedge \mathsf{CHEAT}\right] \\
&\geq \frac{1}{2n^c} \Pr\left[\mathsf{EXTRACT} \mid \mathsf{PREFIX}_{\rho^*} \wedge \mathsf{CHEAT}\right] \qquad (6.5)
\end{aligned}$$

Thus, to show Equation (6.3), it suffices to show that

$$\Pr\left[\mathsf{EXTRACT} \mid \mathsf{PREFIX}_{\rho^*} \wedge \mathsf{CHEAT}\right] \geq \frac{1}{\mathsf{poly}(n)} \quad . \qquad (6.6)$$

Recall that EXTRACT is the event that $\mathcal{B}'$ succeeds in extracting the committed values of the wExtCom commitments from at least $9.9n$ columns. From the construction of wExtCom, EXTRACT occurs if in at least $9.9n$ columns, the challenge bit of the wExtCom commitment on a look-ahead thread is different from the challenge bit on the main thread. Hence, to show Equation (6.6), it suffices to show that in at least $9.9n$ columns, the probability that the challenge bit of wExtCom is b is "high" for both $b = 0$ and $b = 1$ on each look-ahead thread. Furthermore, since each look-ahead thread is generated by repeatedly executing the main thread from ρ^* until a new accepting transcript of Step 3 is obtained, it suffices to show that under the condition that PREFIX_{ρ^*} occurs and Step 3 is accepted, the probability that the challenge bit of the wExtCom commitment is b is "high" for both $b = 0$ and $b = 1$ in at least $9.9n$ columns. Based on these observations, we show the following subclaim.

Subclaim 6.1. *Let ch_j be the random variable representing the challenge bit of* wExtCom *in the j-th column on the main thread, and let* ACCEPT *be the event that every* wExtCom *commitment is accepting on the main thread. Then, there exists a subset $\mathcal{J}_{\text{good}} \subset [10n]$ such that:*

- $|\mathcal{J}_{\text{good}}| \geq 9.9n$
- *For every $j \in \mathcal{J}_{\text{good}}$ and $b \in \{0, 1\}$,*

$$\Pr\left[ch_j = b \mid \mathit{PREFIX}_{\rho^*} \wedge \mathit{ACCEPT}\right] \geq \frac{1}{40n^{c+1}} \quad .$$

Proof. For any $j \in [10n]$ and $b \in \{0, 1\}$, we have

$$\begin{aligned}
\Pr\left[ch_j = b \mid \mathsf{PREFIX}_{\rho^*} \wedge \mathsf{ACCEPT}\right] &= \frac{\Pr\left[\mathsf{ACCEPT} \wedge ch_j = b \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right]}{\Pr\left[\mathsf{ACCEPT} \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right]} \\
&\geq \Pr\left[\mathsf{ACCEPT} \wedge ch_j = b \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right] \quad . \qquad (6.7)
\end{aligned}$$

Hence, we show that in at least $9.9n$ columns, for any $b \in \{0, 1\}$ we have

$$\Pr\left[\mathsf{ACCEPT} \wedge ch_j = b \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right] \geq \frac{1}{40n^{c+1}} \quad . \qquad (6.8)$$

Let

$$\mathcal{J}_{\mathrm{bad}} \stackrel{\mathrm{def}}{=} \left\{ j \in [10n] \;\middle|\; \exists b_j^* \in \{0,1\} \text{ s.t. } \Pr\left[\mathsf{ACCEPT} \wedge ch_j = b_j^* \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right] < \frac{1}{40n^{c+1}} \right\} \ .$$

We have

$$\begin{aligned}
&\Pr\left[\mathsf{ACCEPT} \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right] \\
&\leq \Pr\left[\bigwedge_{j \in \mathcal{J}_{\mathrm{bad}}} ch_j = 1 - b_j^*\right] + \Pr\left[\mathsf{ACCEPT} \bigwedge \left(\bigvee_{j \in \mathcal{J}_{\mathrm{bad}}} ch_j = b_j^*\right) \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right] \\
&\leq 2^{-|\mathcal{J}_{\mathrm{bad}}|} + \sum_{j \in \mathcal{J}_{\mathrm{bad}}} \Pr\left[\mathsf{ACCEPT} \wedge ch_j = b_j^* \mid \mathsf{PREFIX}_{\rho^*}\right] \\
&< 2^{-|\mathcal{J}_{\mathrm{bad}}|} + 10n \cdot \frac{1}{40n^{c+1}} \\
&= 2^{-|\mathcal{J}_{\mathrm{bad}}|} + \frac{1}{4n^c} \ . \qquad (6.9)
\end{aligned}$$

On the other hand, since ACCEPT occurs whenever CHEAT occurs, from Equation (6.4) we have

$$\Pr\left[\mathsf{ACCEPT} \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right] \geq \Pr\left[\mathsf{CHEAT} \;\middle|\; \mathsf{PREFIX}_{\rho^*}\right] \geq \frac{1}{2n^c} \ . \qquad (6.10)$$

From Equations (6.9) and (6.10), we have $|\mathcal{J}_{\mathrm{bad}}| = O(\log n)$ and therefore $|\mathcal{J}_{\mathrm{bad}}| < 0.1n$. Thus, in at least $9.9n$ columns, we have Equation (6.8) for any $b \in \{0,1\}$.

Define $\mathcal{J}_{\mathrm{good}} \stackrel{\mathrm{def}}{=} [10n] \setminus \mathcal{J}_{\mathrm{bad}}$. Since $|\mathcal{J}_{\mathrm{bad}}| < 0.1n$, we have $|\mathcal{J}_{\mathrm{good}}| \geq 9.9n$. Furthermore, from Equations (6.7) and (6.8), for any $j \in \mathcal{J}_{\mathrm{good}}$ and $b \in \{0,1\}$ we have

$$\Pr\left[ch_j = b \mid \mathsf{PREFIX}_{\rho^*} \wedge \mathsf{ACCEPT}\right] \geq \frac{1}{40n^{c+1}} \ .$$

This concludes the proof of Subclaim 6.1. □

As mentioned above, we can obtain Equation (6.1) by using Subclaim 6.1. First, since the distribution of each look-ahead thread is the same as that of the main thread, Subclaim 6.1 implies that under the condition that PREFIX_{ρ^*} and CHEAT occur, $\mathcal{B}'$ requires $40n^{c+1}$ accepted transcripts of Step 3 on average to extract the committed value of wExtCom in the j-th columns for any $j \in \mathcal{J}_{\mathrm{good}}$. Since $\mathcal{B}'$ collects n^{c+3} accepted transcripts, it follows from Markov's inequality that for any $j \in \mathcal{J}_{\mathrm{good}}$, $\mathcal{B}'$ extracts the committed value of wExtCom in the j-th column except with probability $40n^{c+1}/n^{c+3} = 40/n^2$ under the condition that PREFIX_{ρ^*} and CHEAT occur. Thus, from the union bound, $\mathcal{B}'$ extracts the committed value of wExtCom in the j-th column for every $j \in \mathcal{J}_{\mathrm{good}}$ except with probability $9.9n \cdot 40/n^2 = 396/n$. We therefore have

$$\Pr\left[\mathsf{EXTRACT} \mid \mathsf{PREFIX}_{\rho^*} \wedge \mathsf{CHEAT}\right] \geq 1 - \frac{396}{n} \ . \qquad (6.11)$$

Then, from Equations (6.5) and (6.11), we have

$$\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT} \mid \mathsf{PREFIX}_{\rho^*}\right] \geq \frac{1}{2n^c} \cdot \left(1 - \frac{396}{n}\right) \geq \frac{1}{4n^c} \quad . \tag{6.12}$$

Since ρ^* is any prefix in Δ, from Equations (6.2) and (6.12) we have

$$\Pr\left[\mathsf{CHEAT} \wedge \mathsf{EXTRACT}\right] \geq \frac{1}{2n^c} \cdot \frac{1}{4n^c} = \frac{1}{8n^{2c}} \quad .$$

Thus, we have Equation (6.1). We therefore conclude that $\mathcal{B}'$ outputs 1 with probability at least $1/8n^{2c}$ when $\mathcal{B}'$ receives a commitment to Γ_1. Hence, $\mathcal{B}'$ distinguishes a commitment to Γ_1 from a commitment to Γ_0 with advantage $1/8n^{2c} - \mathsf{negl}(n)$.

Now, we are ready to show that $\mathcal{B}$ breaks the hiding property of Com. The running time of $\mathcal{B}$ is clearly at most $\mathsf{poly}(n)$. Hence, to show that $\mathcal{B}$ distinguishes a Com commitment, it suffices to show that the output of $\mathcal{B}$ is the same as that of $\mathcal{B}'$ except with probability $1/n^{2c+1}$. (This is because $\mathcal{B}'$ distinguishes a Com commitment with advantage $1/8n^{2c} - \mathsf{negl}(n)$.) Recall that the output of $\mathcal{B}$ differs from that of $\mathcal{B}'$ if and only if $\mathcal{B}'$ rewinds C^* more than n^{3c+4} times. Let $T(n)$ be a random variable for the number of rewinding in $\mathcal{B}'$. For any prefix ρ of the transcript between C^* and an honest receiver up until the challenge bits of $\mathsf{wExtCom}$ (exclusive), we have

$$\mathrm{E}\left[T(n) \mid \mathsf{PREFIX}_\rho\right] \leq \Pr\left[\mathsf{ACCEPT} \mid \mathsf{PREFIX}_\rho\right] \cdot \frac{n^{c+3}}{\Pr\left[\mathsf{ACCEPT} \mid \mathsf{PREFIX}_\rho\right]} = n^{c+3} \quad .$$

Thus, we have

$$\begin{aligned}\mathrm{E}\left[T(n)\right] &= \sum_\rho \Pr\left[\mathsf{PREFIX}_\rho\right] \mathrm{E}\left[T(n) \mid \mathsf{PREFIX}_\rho\right] \\ &\leq n^{c+3} \sum_\rho \Pr\left[\mathsf{PREFIX}_\rho\right] \leq n^{c+3} \quad .\end{aligned}$$

From Markov's inequality, $\mathcal{B}'$ rewinds C^* more than n^{3c+4} times with probability at most $n^{c+3}/n^{3c+4} = 1/n^{2c+1}$. Thus, the output of $\mathcal{B}$ is the same as that of $\mathcal{B}'$ except with probability $1/n^{2c+1}$, and therefore $\mathcal{B}$ distinguishes a commitment to Γ_1 from a commitment to Γ_0 with advantage at least $1/8n^{2c} - \mathsf{negl}(n) - 1/n^{2c+1} \geq 1/16n^{2c}$. □

Proof of Claim 6.3.

From the binding property of ExtCom, the committed values $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of the ExtCom commitments in Step 2 are uniquely determined except with negligible probability. Hence, to prove the claim, it suffices to show that the following holds in an accepting $\mathsf{sExtCom}$ commitment only with negligible probability.

- The committed values $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of the ExtCom commitments are uniquely determined, but
- $\boldsymbol{s}$ is 0.8-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $s_j = w_j$ for every $j \in \Gamma$, but $\boldsymbol{s}$ is 0.1-far from $\boldsymbol{w}$.

Assume for contradiction that for infinitely many n, the above hold in an accepting $\mathsf{sExtCom}$ commitment with probability at least $1/p(n)$ for a polynomial $p(\cdot)$. Then, from Claim 6.2, the following holds in an accepting $\mathsf{sExtCom}$ commitment with probability at least $1/2p(n)$ for infinitely many n.

- At least $9.5n$ of the ExtCom commitments are valid, and
- the committed values $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of the ExtCom commitments are uniquely determined, but
- $\boldsymbol{s}$ is 0.8-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $s_j = w_j$ for every $j \in \Gamma$, but $\boldsymbol{s}$ is 0.1-far from $\boldsymbol{w}$.

Fix any such n. We derive a contradiction by constructing an adversary $\mathcal{B}$ that breaks the hiding property of Com. For random subsets $\Gamma_0, \Gamma_1 \subset [10n]$ of size n, $\mathcal{B}$ tries to distinguish a Com commitment to Γ_0 from a Com commitment to Γ_1 as follows. $\mathcal{B}$ internally invokes C^* and interacts with it as a receiver of $\mathsf{sExtCom}$ honestly except for the following.

- In Step 1, $\mathcal{B}$ receives a Com commitment from the external committer (who commits to either Γ_0 or Γ_1) and forwards the commitment to C^* as the commitment in Step 1.
- In Step 2, the committed values are extracted by using the extractor of ExtCom. If the extractor runs more than $6p(n) \cdot T(n)$ steps, $\mathcal{B}$ terminates immediately with output fail, where $T(n) = \mathsf{poly}(n)$ is an expected running time of the extractor of ExtCom. Otherwise, let $\widetilde{\boldsymbol{s}} = (\widetilde{s}_1, \ldots, \widetilde{s}_{10n})$ be the extracted values.
- After Step 2 ends, $\mathcal{B}$ outputs 1 if there exists a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ such that $\widetilde{\boldsymbol{s}}$ is 0.8-close to but 0.05-far from $\boldsymbol{w}$ and that $\widetilde{s}_j = w_j$ holds for every $j \in \Gamma_1$. Otherwise, $\mathcal{B}$ outputs 0.

First, we analyze an adversary $\mathcal{B}'$ that is the same as $\mathcal{B}$ except that $\mathcal{B}'$ does not terminate even after the extractor of ExtCom runs more than $6p(n) \cdot T(n)$ steps. When $\mathcal{B}'$ receives a commitment to Γ_0, the internal C^* receives no information about Γ_1, so the probability that $\widetilde{\boldsymbol{s}}$ is 0.05-far from $\boldsymbol{w}$ but $\widetilde{s}_j = w_j$ holds for every $j \in \Gamma_1$ is exponentially small; thus, $\mathcal{B}'$ outputs 1 with exponentially small probability. We next compute the probability that $\mathcal{B}'$ outputs 1 when it receives a commitment to Γ_1. From our assumption, with probability $1/2p(n)$ it holds that $9.5n$ of the ExtCom commitments are valid and the unique committed values $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of the ExtCom commitments are 0.8-close to but 0.1-far from a valid codeword $\boldsymbol{w}$ that satisfies $s_j = w_j$ for every $j \in \Gamma_1$. Since the extractability of ExtCom guarantees that $\widetilde{s}_j = s_j$ holds except with negligible probability when the j-th ExtCom commitment is valid (and in particular when $s_j = w_j \neq \bot$), with probability at least $1/3p(n)$, $\widetilde{\boldsymbol{s}}$ is 0.8-close to but 0.05-far from a valid codeword $\boldsymbol{w}$ that satisfies $\widetilde{s}_j = w_j$ for every $j \in \Gamma_1$. Hence, when $\mathcal{B}'$ receives a commitment to Γ_1, $\mathcal{B}'$ outputs 1 with probability at least $1/3p(n)$. Therefore, $\mathcal{B}'$ distinguishes a Com commitment with advantage $1/3p(n) - \mathsf{negl}(n)$.

Now, we are ready to argue that $\mathcal{B}$ breaks the hiding property of Com. The output of $\mathcal{B}$ differs from that of $\mathcal{B}'$ if and only if the extraction from ExtCom takes more than

$6p(n) \cdot T(n)$ steps. From Markov's inequality, the extraction from ExtCom takes more than $6p(n) \cdot T(n)$ steps only with probability $1/6p(n)$. Hence, $\mathcal{B}$ distinguishes a Com commitment with advantage $1/3p(n) - \mathsf{negl}(n) - 1/6p(n) \geq 1/6p(n) - \mathsf{negl}(n)$. Since the running time of $\mathcal{B}$ can be bounded by $\mathsf{poly}(n)$, $\mathcal{B}$ breaks the hiding property of Com. □

Conclusion of Proof of Lemma 6.2.

From Claims 6.2 and 6.3, the probability that the ExtCom commitments are not good in an accepting $\mathsf{sExtCom}$ commitment is negligible. Hence, from Claim 6.1, the extractor E outputs a correct committed value except with negligible probability. This completes the proof of Lemma 6.2. □

6.4.2 One-One CCA-Secure Commitment Scheme

Using one-way functions in a black-box way, we construct a $O(\log n)$-round one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$. Recall that a commitment scheme is one-one CCA secure if it is CCA secure w.r.t. a restricted class of adversaries that start only a single right session. Our scheme does not satisfy the statistically binding property but does satisfy the strong computational binding property.

Lemma 6.3. *Assume the existence of one-way functions. Then, there exists a $O(\log n)$-round one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$ that satisfies the strong computational binding property and the computational hiding property. Furthermore, $\mathsf{CCACom}^{1:1}$ uses the underlying one-way function only in a black-box way.*

Proof. We construct $\mathsf{CCACom}^{1:1}$ by slightly modifying the black-box $O(n^{\epsilon})$-round CCA-secure commitment scheme of Lin and Pass [LP12] and then applying the "DDN $\log n$ trick" [DDN00, LPV08] on it, where the DDN $\log n$ trick is a transformation by Dolev, Dwork, and Naor (DDN) [DDN00] and has been used to transform concurrent non-malleable commitment schemes for tags of length $O(\log n)$ to non-malleable commitment schemes for tags of length $O(n)$ without increasing round complexity.

First, we recall the CCA-secure commitment scheme of [LP12] (see Figure 6.5). Roughly speaking, the commitment scheme of [LP12] consists of $4\ell(n)\eta(n)$ *rows*—each row is a parallel execution of a part of the trapdoor commitment scheme $\mathsf{PTrapCom}$ of [PW09] (see Section 6.3.4)—followed by a cut-and-choose phase, where $\ell(n)$ is the length of the tag and $\eta(n) \stackrel{\text{def}}{=} n^{\epsilon}$ for $\epsilon > 0$. In the analysis of [LP12], which is based on that of [CLP10, CLP16], it is shown that in any transcript of one left session and many right sessions of the scheme, each right session has $\Omega(\eta(n))$ $\mathsf{safe\text{-}points}$, from which we can rewind the right session and extract its committed value without breaking the hiding property of the left session. Then, since each right session has $\Omega(\eta(n))$ $\mathsf{safe\text{-}points}$, we can extract the committed value of each right session even in the concurrent setting by using the rewinding strategy of Richardson and Kilian [RK99] to deal with the problem of recursive rewinding. Thus, by extracting the committed-value of a row in each right session, we can emulate the committed-value oracle in polynomial time without breaking the hiding property of the left session. Thus, the CCA security follows from the hiding property of the left session.

Commit Phase

Let ℓ, η be two polynomials such that $\ell(n) = n^{\nu}$ and $\eta(n) = n^{\epsilon}$ for $\nu, \epsilon > 0$, and L be a polynomial such that $L(n) = 4\ell(n)\eta(n)$. To commit to a value v, the committer C and the receiver R, on common input 1^n and $\mathsf{id} \in \{0,1\}^{\ell(n)}$, do the following.

Stage 1: R sends the Step 1 message of a commitment of PTrapCom. That is, a commitment of Com to a randomly chosen string challenge $\boldsymbol{e} = (e_1, \ldots, e_n)$.

Stage 2: C computes an $(n+1)$-out-of-$10n$ Shamir's secret sharing $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of value v, and commits to these shares using Step 2 of PTrapCom in parallel for $L(n)$ times; we call the i-th parallel commitment the i-th *row*, and all the commitments to s_j the j-th *column*. Messages in the $4\ell(n)\eta(n)$ rows are scheduled based on id and the schedules design_0 and design_1 depicted in Figure 6.6. More precisely, Stage 2 consists of $\ell(n)$ phases. In phase i, C provides $\eta(n)$ sequential $\mathsf{design}_{\mathsf{id}_i}$ pairs of rows, followed by $\eta(n)$ sequential $\mathsf{design}_{1-\mathsf{id}_i}$ pairs of rows.

Stage 3: R decommits the commitment in Stage 1 to $\boldsymbol{e}$. C completes the $10nL(n)$ executions of PTrapCom w.r.t. challenge $\boldsymbol{e}$ in parallel.

Stage 4: R sends a randomly chosen subset $\Gamma \subset [10n]$ of size n. For every $j \in \Gamma$, C decommits all the commitments in the j-th column. R checks whether all the decommitments are valid and reveal the same committed values s_j.

Figure 6.5: Black-box CCA-secure commitment scheme of [LP12]

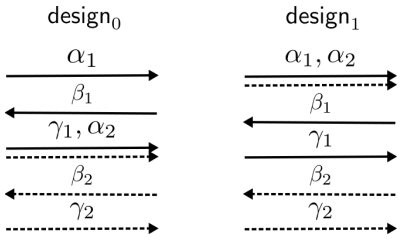

Figure 6.6: Description of the schedules used in Stage 2 of the protocol of [LP12]. $(\alpha_1, \beta_1, \gamma_1)$ and $(\alpha_2, \beta_2, \gamma_2)$ are the transcripts of a pair of rows in Stage 2.

Next, we observe that by setting $\eta(n) := 1$ in the scheme of [LP12], we obtain a black-box $O(\ell(n))$-round *parallel CCA-secure* commitment scheme for tags of length $\ell(n)$, where a commitment scheme is *parallel CCA secure* if it is CCA secure w.r.t. a restricted class of adversaries that start only a single parallel right session. This is because when an adversary starts only a single parallel right session, the problem of recursive rewinding does not occur, so each right session need to have only a single safe-point as in the concurrent non-malleable commitment scheme of [LPV08] (on which the CCA-secure commitment schemes of [CLP10, CLP16, LP12] are based). Therefore, by setting $\eta(n) := 1$ and $\ell(n) := O(\log n)$, we obtain a black-box $O(\log n)$-round commitment scheme that is parallel CCA secure for tags of length $O(\log n)$.

We then observe that the DDN $\log n$ trick [DDN00, LPV08] transforms any black-box parallel CCA-secure commitment scheme for tags of length $O(\log n)$ to a black-box one-one CCA-secure commitment scheme for tags of length $O(n)$. This can be proven in essentially the same way as the proof of the fact that the DDN $\log n$ trick transforms a concurrent non-malleable commitment scheme for tags of length $O(\log n)$ to a non-malleable commitment scheme for tags of length $O(n)$. For details, see Section 6.7.1.

Combining the above, we obtain a black-box $O(\log n)$-round one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$. $\mathsf{CCACom}^{1:1}$ satisfies the strong computational binding property and the computational hiding property because the CCA-secure commitment scheme of [LP12] satisfies both properties and the DDN $\log n$ trick preserves both properties. (The strong computational binding property of [LP12] follows from that of the trapdoor commitment scheme of [PW09].) □

6.5 CCA-Secure Commitment Scheme

In this section, we construct a $\widetilde{O}(\log^2 n)$-round robust CCA-secure commitment scheme by using one-way functions in a black-box way.

Theorem 6.1. *Assume the existence of one-way functions. Then, there exists a* $\widetilde{O}(\log^2 n)$*-round robust CCA-secure commitment scheme* CCACom. *Furthermore,* CCACom *uses the underlying one-way function only in a black-box way.*

Proof. CCACom is shown in Figure 6.7, in which we use the following tools (all of which can be constructed from one-way functions in a black-box way).

- A two-round statistically binding commitment scheme Com. (A concrete example is Naor's commitment scheme in Section 2.3.1, which can be constructed from one-way functions in a black-box way [Nao91, HILL99].)
- The concurrently extractable commitment scheme CECom of Micciancio et al. [MOSV06]. (See Section 2.3.3.) The parameter ℓ in CECom is set as $\ell := O(\log^2 n \log\log n)$ so that $\ell = \omega(\log^2 n)$.
- A constant-round strongly extractable commitment scheme sExtCom. (See Lemma 6.2 in Section 6.4.1.)

- A $O(\log n)$-round one-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$ that satisfies strong computational binding property. (See Lemma 6.3 in Section 6.4.2.)

The round complexity of CCACom is clearly $\widetilde{O}(\log^2 n)$. The statistical binding property of CCACom follows directly from that of Com. Hence, it remains to show that CCACom is robust CCA secure. (The hiding property follows from CCA security.) In what follows, we prove CCA security in Section 6.5.1 and robustness in Section 6.5.2.

6.5.1 Proof of CCA Security

Lemma 6.4. CCACom *is CCA secure.*

Proof. We first remark that from the construction of the decommit phase of CCACom, the committed value of a CCACom commitment is defined as follows.

Definition 6.4 (Committed value of CCACom)**.** *If the shares* $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ *that are committed to in Stage 2 are* 0.9*-close to a valid codeword* $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ *that satisfies* $w_j = s_j$ *for every* $j \in \Gamma$*, the committed value of a* CCACom *commitment is* $\mathsf{Decode}(\boldsymbol{w})$*. Otherwise, the committed value is* $\perp$ *(i.e., the commitment is invalid).* ◇

We notice that the function $\mathsf{Value}_\Gamma(\cdot)$ in Figure 6.1 (Section 6.3.1) computes the committed value of a CCACom commitment as above on input the shares $\boldsymbol{s}$ that are committed to in Stage 2.

To prove the CCA security of CCACom, we show the following indistinguishability for any PPT adversary $\mathcal{A}_{\mathrm{cca}}$ (cf. Definition 2.7).

$$\{\mathsf{IND}_0(\mathsf{CCACom}, \mathcal{A}_{\mathrm{cca}}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \{\mathsf{IND}_1(\mathsf{CCACom}, \mathcal{A}_{\mathrm{cca}}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*} . \tag{6.13}$$

Fix any PPT adversary $\mathcal{A}_{\mathrm{cca}}$. We prove Indistinguishability (6.13) by a hybrid argument. Since the experiments $\mathrm{IND}_0(\mathsf{CCACom}, \mathcal{A}_{\mathrm{cca}}, n, z)$ and $\mathrm{IND}_1(\mathsf{CCACom}, \mathcal{A}_{\mathrm{cca}}, n, z)$ differ only in the value that is committed to in the left session, we consider a series of hybrid experiments in which the left session is gradually modified so that in the last hybrid the adversary receives no information about the value that is committed to in the left session. Formally, for each $b \in \{0, 1\}$, we consider the following hybrid experiments.

Hybrid $H_0^b(n, z)$: Hybrid $H_0^b(n, z)$ is the same as $\mathrm{IND}_b(\mathsf{CCACom}, \mathcal{A}_{\mathrm{cca}}, n, z)$.

Hybrid $H_1^b(n, z)$ to Hybrid $H_\eta^b(n, z)$: For $k \in [\eta]$, Hybrid $H_k^b(n, z)$ is the same as $H_0^b(n, z)$ except for the following.

- In Stage 1 of the left session, the committed value Γ is extracted by brute force from the $\mathsf{CCACom}^{1:1}$ commitment. If the commitment is invalid, Γ is set to be a random subset. If the commitment has more than one committed value, $H_k^b(n, z)$ outputs fail and terminates.

Commit Phase

The committer C and the receiver R take common inputs 1^n and $\mathsf{id} \in \{0,1\}^n$, and C additionally takes private input $v \in \{0,1\}^n$. To commit to v, the committer C does the following with the receiver R.

Stage 1. R commits to a random subset $\Gamma \subset [10n]$ of size n by using $\mathsf{CCACom}^{1:1}$ with tag id.

Stage 2. C computes an $(n+1)$-out-of-$10n$ Shamir's secret sharing $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of value v. Next, for each $j \in [10n]$ in parallel, C commits to s_j by using Com. Let $\phi_1, \ldots, \phi_{10n}$ denote the commitments and $d_1, \ldots, d_{10n}$ denote the decommitments.

Stage 3. For each $j \in [10n]$ in parallel, C commits to s_j by using CECom, where the parameter ℓ in CECom is set as $\ell := \widetilde{O}(\log^2 n)$. Let $\psi_1, \ldots, \psi_{10n}$ denote the commitments and $e_1, \ldots, e_{10n}$ denote the decommitments.

We call these parallel CECom commitments the *row* of CECom, and the commitment ψ_j in the row the *j-th column* (of CECom).

Stage 4. Let $R_{\mathsf{CCA}^{1:1}} := R_{\mathsf{CCA}^{1:1}}(n)$ be the round complexity of $\mathsf{CCACom}^{1:1}$. Let $\eta := R_{\mathsf{CCA}^{1:1}} + 1$. Then, for each $i \in [\eta]$ in sequence, C does the following.

- For each $j \in [10n]$ in parallel, C commits to (s_j, d_j, e_j) by using $\mathsf{sExtCom}$. Let $\theta_{i,1}, \ldots, \theta_{i,10n}$ denote the commitments.

We call the i-th parallel commitments the *i-th row* (of $\mathsf{sExtCom}$), and call all the commitments to (s_j, d_j) the *j-th column* (of $\mathsf{sExtCom}$).

Stage 5. R decommits the commitment in Stage 1 to Γ.

Stage 6. For each $j \in \Gamma$, C decommits $\theta_{i,j}$ to (s_j, d_j, e_j) for each $i \in [\eta]$. Then, for every $j \in \Gamma$, R checks whether (s_j, d_j) is a valid decommitment of ϕ_j and (s_j, e_j) is a valid decommitment of ψ_j.

Decommit Phase

- C sends v and decommits the Com commitments $\phi_1, \ldots \phi_{10n}$ in Stage 2 to $s_1, \ldots, s_{10n}$.
- R accepts if and only if the following holds w.r.t. $\boldsymbol{s} = (s_1, \ldots, s_{10n})$, where s_j is defined to be $\perp$ if the decommitment of ϕ_j is invalid.
  - $\boldsymbol{s}$ is 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_j = s_j$ for every $j \in \Gamma$, and $\boldsymbol{w}$ is a codeword of v.

Figure 6.7: CCA commitment scheme CCACom.

- In Stage 4 of the left session, the left committer commits to $0^{|u_j|}$ instead of u_j for every $j \notin \Gamma$ in the i-th row for $i \in [k]$.

Hybrid $H^b_{\eta+1}(n,z)$: Hybrid $H^b_{\eta+1}(n,z)$ is the same as $H^b_{\eta}(n,z)$ except that in Stage 3 of the left session, the left committer commits to $0^{|s_j|}$ instead of s_j for every $j \notin \Gamma$.

Hybrid $H^b_{\eta+2}(n,z)$: Hybrid $H^b_{\eta+2}(n,z)$ is the same as $H^b_{\eta+1}(n,z)$ except that in Stage 2 of the left session, the left committer commits to $0^{|s_j|}$ instead of s_j for every $j \notin \Gamma$.

For $k \in \{0,\dots,\eta+2\}$, let $\mathsf{H}^b_k(n,z)$ be the random variable for the output of $H^b_k(n,z)$. Since $\mathcal{A}_{\mathrm{cca}}$ receives no information about b in $H^b_{\eta+2}(n,z)$, we have

$$\left\{\mathsf{H}^0_{\eta+2}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} = \left\{\mathsf{H}^1_{\eta+2}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} . \tag{6.14}$$

Hence, from a hybrid argument, we can show Indistinguishability (6.13) by showing the following three claims.

Claim 6.4. *For every $b \in \{0,1\}$ and $k \in [\eta]$, we have the following indistinguishability.*

$$\left\{\mathsf{H}^b_{k-1}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{H}^b_{k}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} .$$

Claim 6.5. *For every $b \in \{0,1\}$, we have the following indistinguishability.*

$$\left\{\mathsf{H}^b_{\eta}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{H}^b_{\eta+1}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} .$$

Claim 6.6. *For every $b \in \{0,1\}$, we have the following indistinguishability.*

$$\left\{\mathsf{H}^b_{\eta+1}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{H}^b_{\eta+2}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} .$$

We prove Claim 6.4 in Section 6.5.1.1 and prove Claims 6.5 and 6.6 in Section 6.5.1.4.

6.5.1.1 Proof of Claim 6.4

Below, we prove Claim 6.4 using the following subclaim.

Subclaim 6.2. *For every $b \in \{0,1\}$ and $k \in \{0,\dots,\eta+2\}$, $H^b_k(n,z)$ outputs* fail *with at most negligible probability.*

The proof of Subclaim 6.2 is given in Section 6.5.1.3.

Proof of Claim 6.4. Since $H^b_{k-1}(n,z)$ and $H^b_k(n,z)$ differ only in the values that are committed to in a row of sExtCom in the left session, we use the hiding property of sExtCom to prove the indistinguishability. A problem is that $\mathcal{A}_{\mathrm{cca}}$ interacts with the committed-value oracle O, which runs in super-polynomial time; because of the super-polynomial-time power of O, the indistinguishability between the two hybrids does not follow directly from the *computational* hiding property of sExtCom. To overcome this problem, we show that the oracle O can be emulated in polynomial time. Specifically, we show that the oracle O can be emulated by extracting the shares that are committed to in the rows of CECom and then computing the committed values of the right sessions from the extracted shares. When extracting the committed shares from the row of

CECom, we use the robust concurrent extraction lemma (Lemma 2.1) so that we can use the hiding property of the k-th row of sExtCom even in the presence of the extraction from CECom. Formally, we consider the following hybrids $G^b_{h:1}(n,z),\ldots,G^b_{h:3}(n,z)$ for each $h \in \{k-1,k\}$.

Hybrid $G^b_{h:1}(n,z)$: Hybrid $G^b_{h:1}(n,z)$ is the same as $H^b_h(n,z)$ except that at the end of each right session, the oracle O returns $\mathsf{Value}_\Gamma(s^{\mathrm{CEC}})$ to $\mathcal{A}_{\mathrm{cca}}$ rather than $\mathsf{Value}_\Gamma(s)$ as the committed value of this session, where $s=(s_1,\ldots,s_{10n})$ is the shares that are committed to in the row of Com in Stage 2, $s^{\mathrm{CEC}}=(s^{\mathrm{CEC}}_1,\ldots,s^{\mathrm{CEC}}_{10n})$ is the shares that are committed to in the row of CECom in Stage 3, and Γ is the subset that is committed to in the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1.

Hybrid $G^b_{h:2}(n,z)$: Hybrid $G^b_{h:2}(n,z)$ is the same as $G^b_{h:1}(n,z)$ except for syntactical differences: Roughly speaking, $G^b_{h:2}(n,z)$ is an experiment in which $G^b_{h:1}(n,z)$ is executed in such a way that we can use the robust concurrent extraction lemma (Lemma 2.1) later. Formally, $G^b_{h:2}(n,z)$ is defined as follows. Recall that in the setting of the robust concurrent extraction lemma, an adversary, $\mathcal{A}_{\mathrm{robust}}$, launches the robust-concurrent attack by interacting with the online extractor $\mathcal{E}$; specifically $\mathcal{A}_{\mathrm{robust}}$ interacts with $\mathcal{E}$ as a party A of an arbitrary two-party protocol $\Pi=\langle B,A\rangle$ while interacting with $\mathcal{E}$ as the committers of CECom concurrently and obtaining a value from $\mathcal{E}$ at the end of each session of CECom (where the values that are returned from $\mathcal{E}$ are supposed to be the committed values of the CECom sessions). Then, consider the following Π and $\mathcal{A}_{\mathrm{robust}}$ (see also Figure 6.8).

$\Pi=\langle B,A\rangle$**:** First, party A gives a $\mathsf{CCACom}^{1:1}$ commitment to party B, where the tag in the $\mathsf{CCACom}^{1:1}$ commitment is chosen by A. Then, B extracts the committed value Γ of this $\mathsf{CCACom}^{1:1}$ commitment by brute force and sends it back to A. (If the $\mathsf{CCACom}^{1:1}$ commitment is invalid, Γ is set to be a random subset, and if the $\mathsf{CCACom}^{1:1}$ commitment has more than one committed value, B outputs fail and terminates.)

Next, A sends a sequence of strings $(m_1,\ldots,m_{9n})$ to B. Then, when $h=k-1$, B commits to each m_j ($j\in[9n]$) in parallel using sExtCom, and when $h=k$, B commits to each $0^{|m_j|}$ ($j\in[9n]$) in parallel using sExtCom.

$\mathcal{A}_{\mathrm{robust}}$**:** $\mathcal{A}_{\mathrm{robust}}$ takes non-uniform advice z and internally executes $G^b_{h:1}(n,z)$ with the following changes. (Recall that the execution of $G^b_{h:1}(n,z)$ involves an interaction with the CCA-security adversary $\mathcal{A}_{\mathrm{cca}}$.)

- In Stage 1 of the left session, $\mathcal{A}_{\mathrm{robust}}$ forwards the $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{A}_{\mathrm{cca}}$ to the online extractor $\mathcal{E}$ (who internally emulates party B of Π). Then, instead of extracting the committed subset Γ from this $\mathsf{CCACom}^{1:1}$ commitment by brute force, $\mathcal{A}_{\mathrm{robust}}$ obtains Γ from $\mathcal{E}$.
- In the k-th row of sExtCom of the left session, $\mathcal{A}_{\mathrm{robust}}$ sends $\{u_j\}_{j\notin\Gamma}$ to $\mathcal{E}$ (who internally emulates party B of Π), receives sExtCom commitments from $\mathcal{E}$, and forwards them to $\mathcal{A}_{\mathrm{cca}}$. (At the same time, $\mathcal{A}_{\mathrm{robust}}$ correctly commits to $\{u_j\}_{j\in\Gamma}$ for $\mathcal{A}_{\mathrm{cca}}$ by using sExtCom.)

- In Stage 3 of each right session, $\mathcal{A}_{\text{robust}}$ receives a row of CECom commitments from $\mathcal{A}_{\text{cca}}$ and forwards it to $\mathcal{E}$ (who internally emulates the receivers of CECom). Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ denote the responses from $\mathcal{E}$ at the end of the row of the CECom commitments.
- At the end of each right session, $\mathcal{A}_{\text{robust}}$ sends $\mathsf{Value}_\Gamma(\boldsymbol{\alpha})$ to $\mathcal{A}_{\text{cca}}$ as the committed value of this right session.

The output of $\mathcal{A}_{\text{robust}}$ is that of the internally executed $G^b_{h:1}(n, z)$.

From the robust concurrent extraction lemma, there exists a robust simulator $\mathcal{S}$ such that for the above $\mathcal{A}_{\text{robust}}$, there exists an online extractor $\mathcal{E}$ that satisfies the following.

- For any row of CECom that $\mathcal{A}_{\text{robust}}$ sends to $\mathcal{E}$, let $\boldsymbol{s}^{\text{CEC}} = (s_1^{\text{CEC}}, \ldots, s_{10n}^{\text{CEC}})$ be the shares that are committed to in this row of CECom and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ be the responses from $\mathcal{E}$ at the end of this row. Then, for every $j \in [10n]$, if the j-th CECom commitment in this row is valid and its committed value is uniquely determined, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ satisfies $\alpha_j = s_j^{\text{CEC}}$.
- $\mathcal{S}$ can simulate the robust-concurrent attack between $\mathcal{A}_{\text{robust}}$ and $\mathcal{E}$.

Hybrid $G^b_{h:2}(n, z)$ is the experiment $\text{REAL}^{\mathcal{A}_{\text{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$ of the robust concurrent extraction lemma. The output of $G^b_{h:2}(n, z)$ is that of $\mathcal{A}_{\text{robust}}$ in $\text{REAL}^{\mathcal{A}_{\text{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$.

Hybrid $G^b_{h:3}(n, z)$: Hybrid $G^b_{h:3}(n, z)$ differs from $G^b_{h:2}(n, z)$ in that the execution of $\text{REAL}^{\mathcal{A}_{\text{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$ (i.e., the robust-concurrent attack between $\mathcal{A}_{\text{robust}}$ and $\mathcal{E}$) is replaced with an interaction between party B of Π and the robust simulator $\mathcal{S}$ of the robust concurrent extraction lemma (see Figure 6.9). The output of $G^b_{h:3}(n, z)$ is that of $\mathcal{A}_{\text{robust}}$ that is simulated by $\mathcal{S}$.

For $\ell \in \{1, 2, 3\}$, let $\mathsf{G}^b_{h:\ell}(n, z)$ be the random variable for the output of $G^b_{h:\ell}(n, z)$. We now prove the following four claims.

Claim 6.7. *For every $b \in \{0, 1\}$ and $h \in \{k-1, k\}$, we have the following indistinguishability.*

$$\left\{\mathsf{H}^b_h(n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{s}{\approx} \left\{\mathsf{G}^b_{h:1}(n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} .$$

Claim 6.8. *For every $b \in \{0, 1\}$ and $h \in \{k-1, k\}$, we have the following indistinguishability.*

$$\left\{\mathsf{G}^b_{h:1}(n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{s}{\approx} \left\{\mathsf{G}^b_{h:2}(n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} .$$

Claim 6.9. *For every $b \in \{0, 1\}$ and $h \in \{k-1, k\}$, we have the following indistinguishability.*

$$\left\{\mathsf{G}^b_{h:2}(n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{s}{\approx} \left\{\mathsf{G}^b_{h:3}(n, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} .$$

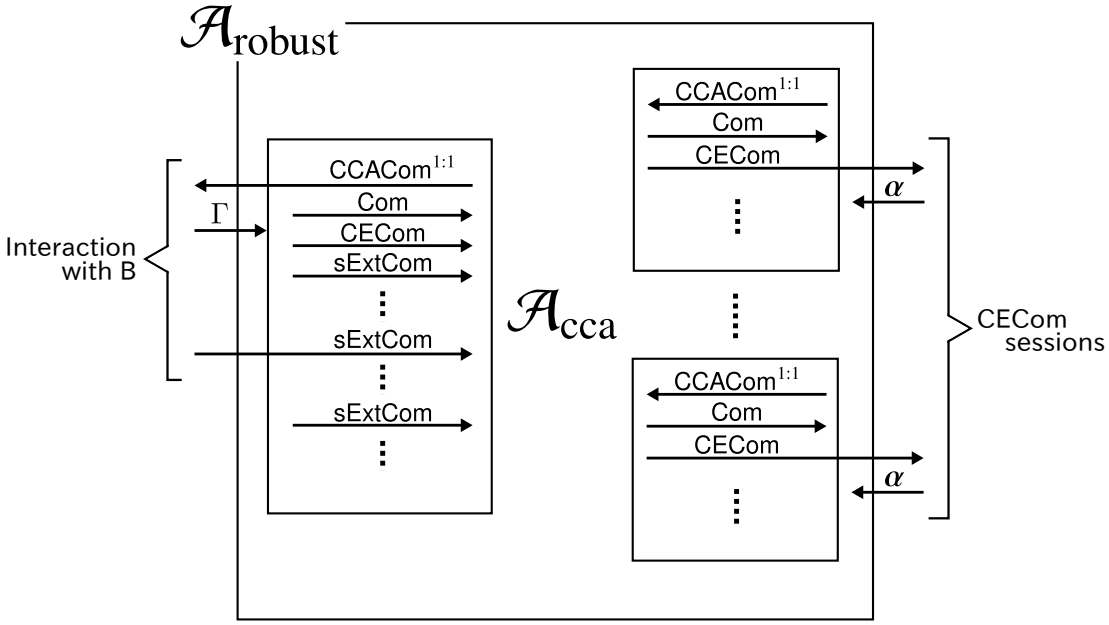

Figure 6.8: Adversary $\mathcal{A}_{\text{robust}}$ in Hybrid $G^b_{h:2}(n,z)$. For simplicity, the right sessions are illustrated as if they are executed sequentially.

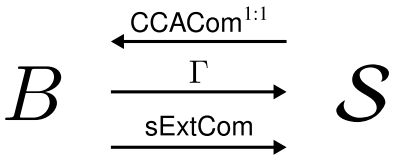

Figure 6.9: Simulator $\mathcal{S}$ in Hybrid $G^b_{h:3}(n,z)$.

Claim 6.10. *For every $b \in \{0,1\}$, we have the following indistinguishability.*

$$\left\{\mathsf{G}^b_{k-1:3}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{G}^b_{k:3}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} .$$

Claim 6.4 follows from these four claims.

Proof of Claim 6.7. Recall that $G^b_{h:1}(n,z)$ differs from $H^b_h(n,z)$ in that the committed value of a right session is computed by $\mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ rather than by $\mathsf{Value}_\Gamma(\boldsymbol{s})$, where $\boldsymbol{s} = (s_1,\ldots,s_{10n})$ is the shares that are committed to in the row of Com in Stage 2, $\boldsymbol{s}^{\mathrm{CEC}} = (s^{\mathrm{CEC}}_1,\ldots,s^{\mathrm{CEC}}_{10n})$ is the shares that are committed to in the row of CECom in Stage 3, and Γ is the subset that is committed to in the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1. Roughly speaking, we prove this claim in two steps.

Step 1. Showing that $\mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}}) = \mathsf{Value}_\Gamma(\boldsymbol{s})$ holds in any right session if $\mathcal{A}_{\mathrm{cca}}$ does not "cheat" in that right session.

Step 2. Showing that $\mathcal{A}_{\mathrm{cca}}$ "cheats" in a right session with at most negligible probability.

Here, we say that $\mathcal{A}_{\mathrm{cca}}$ cheats in a right session if, roughly speaking, in every row of $\mathsf{sExtCom}$ in that session $\mathcal{A}_{\mathrm{cca}}$ does not commit to $u_j = (s_j, d_j, e_j)$ correctly in many columns. Hence, if $\mathcal{A}_{\mathrm{cca}}$ does not cheat, there exists a row of $\mathsf{sExtCom}$ in which $\mathcal{A}_{\mathrm{cca}}$ commits to $u_j = (s_j, d_j, e_j)$ as specified by the protocol in most columns, which guarantees that in most columns the share that is committed to by CECom is equal to the share that is committed to by Com, which in turn guarantees that the committed value of the session can be recovered from the shares that are committed to in the row of CECom instead of from those that are committed to in the row of Com. Details are given below.

First, we define the cheating behavior of $\mathcal{A}_{\mathrm{cca}}$.

Definition 6.5 (Cheating by $\mathcal{A}_{\mathrm{cca}}$)**.** *In each right session, let us say that a row of* $\mathsf{sExtCom}$ *in Stage 4 is* ***bad*** *if the values $\{u'_j = (s'_j, d'_j, e'_j)\}_{j\in[10n]}$ that are committed to in it satisfy the following condition.*

Badness Condition: *Let $\boldsymbol{s}^{sExt} = (s^{sExt}_1,\ldots,s^{sExt}_{10n})$ be the shares that are defined as follows. Let $s^{sExt}_j \overset{\mathrm{def}}{=} s'_j$ if (s'_j, d'_j) is a valid decommitment of the j-th* Com *commitment in Stage 2 and (s'_j, e'_j) is a valid decommitment of the j-th* CECom *commitment in Stage 3. Let $s^{sExt}_j \overset{\mathrm{def}}{=} \perp$ otherwise. Then, the badness condition is defined as follows.*

1. $\left|\left\{j \in [10n] \text{ s.t. } s^{sExt}_j = \perp\right\}\right| \geq n \wedge \left\{j \in [10n] \text{ s.t. } s^{sExt}_j = \perp\right\} \cap \Gamma = \emptyset$, *or*
2. $\boldsymbol{s}^{sExt}$ *is* 0.8*-close to a valid codeword $\boldsymbol{w} = (w_1,\ldots,w_{10n})$ that satisfies $w_j = s^{sExt}_j$ for every $j \in \Gamma$, but $\boldsymbol{s}^{sExt}$ is* 0.1*-far from $\boldsymbol{w}$.*

Let us say that a row of $\mathsf{sExtCom}$ *is* ***good*** *if it is not bad. Then, we say that $\mathcal{A}_{\mathrm{cca}}$* ***cheats*** *in a right session if every row of* $\mathsf{sExtCom}$ *in that right session is bad.* ◇

We then prove the following two subclaims.

Subclaim 6.3. *If the probability that $\mathcal{A}_{\mathrm{cca}}$ cheats in a right session in $H^b_h(n,z)$ is negligible, we have the following indistinguishability.*

$$\left\{\mathsf{H}^b_h(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} \stackrel{s}{\approx} \left\{\mathsf{G}^b_{h:1}(n,z)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} .$$

Subclaim 6.4. *The probability that $\mathcal{A}_{\mathrm{cca}}$ cheats in a right session in $H^b_h(n,z)$ is negligible.*

The proof of Subclaim 6.3 is given below. The proof of Subclaim 6.4 is given in Section 6.5.1.2.

Proof of Subclaim 6.3. We first show that $\mathsf{Value}_\Gamma(\boldsymbol{s}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ holds in an accepting right session if $\mathcal{A}_{\mathrm{cca}}$ does not cheat in that right session, where, as defined in the description of Hybrid $G^b_{h:1}(n,z)$, $\boldsymbol{s} = (s_1,\ldots,s_{10n})$ is the shares that are committed to in the row of Com in Stage 2, $\boldsymbol{s}^{\mathrm{CEC}} = (s^{\mathrm{CEC}}_1,\ldots,s^{\mathrm{CEC}}_{10n})$ is the shares that are committed to in the row of CECom in Stage 3, and Γ is the subset that is committed to in the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1. Fix any right session, and assume that that right session is accepting and $\mathcal{A}$ does not cheat in it. Then, from the definition of cheating (Definition 6.5), that right session has a good row of $\mathsf{sExtCom}$. Let $\{u'_j = (s'_j, d'_j, e'_j)\}_{j\in[10n]}$ be the values that are committed to in that good row of $\mathsf{sExtCom}$. Let $\boldsymbol{s}^{\mathrm{sExt}} = (s^{\mathrm{sExt}}_1,\ldots,s^{\mathrm{sExt}}_{10n})$ be the shares that are derived from $\boldsymbol{u'} = (u'_1,\ldots,u'_{10n})$ as in the definition of cheating. Then, from the definitions of cheating and $\boldsymbol{s}^{\mathrm{sExt}}$, we have the following.

1. For every $j\in[10n]$, if $s^{\mathrm{sExt}}_j \neq \bot$, it holds $s^{\mathrm{sExt}}_j = s_j = s^{\mathrm{CEC}}_j$.
   (This follows from the definition of $\boldsymbol{s}^{\mathrm{sExt}}$.)
2. $\left|\left\{j \text{ s.t. } s^{\mathrm{sExt}}_j = \bot\right\}\right| < n \wedge \left\{j \text{ s.t. } s^{\mathrm{sExt}}_j = \bot\right\} \cap \Gamma = \emptyset$.
   (This is because the session would be rejected in Stage 6 if $\{j \text{ s.t. } s^{\mathrm{sExt}}_j = \bot\} \cap \Gamma \neq \emptyset$.)
3. $\boldsymbol{s}^{\mathrm{sExt}}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_1,\ldots,w_{10n})$ that satisfies $w_j = s^{\mathrm{sExt}}_j$ for every $j\in\Gamma$ or 0.2-far from any such valid codeword.

Hence, from Lemma 6.1 in Section 6.3.1, we have $\mathsf{Value}_\Gamma(\boldsymbol{s}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{sExt}})$ in that session. Therefore, for any accepting right session, we have $\mathsf{Value}_\Gamma(\boldsymbol{s}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ if $\mathcal{A}_{\mathrm{cca}}$ does not cheat in that session.

Since $G^b_{h:1}(n,z)$ differs from $H^b_h(n,z)$ only in that O returns $\mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ to $\mathcal{A}_{\mathrm{cca}}$ rather than $\mathsf{Value}_\Gamma(\boldsymbol{s})$ in each right session, we conclude that $\mathsf{H}^b_h(n,z)$ and $\mathsf{G}^b_{h:1}(n,z)$ are statistically indistinguishable if $\mathcal{A}_{\mathrm{cca}}$ cheats in a right session with at most negligible probability. □

Now, Claim 6.7 follows immediately from Subclaims 6.3 and 6.4. This concludes the proof of Claim 6.7. □

Proof of Claim 6.8. From the construction of $G^b_{h:2}(n,z)$, the execution of $G^b_{h:1}(n,z)$ is perfectly emulated in $G_{h:2}(n,z)$ as long as we have $\mathsf{Value}_\Gamma(\boldsymbol{\alpha}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ in each accepting right session.

First, we observe that if $\mathcal{A}_{\mathrm{cca}}$ does not cheat in an accepting right session, we have $\mathsf{Value}_\Gamma(\boldsymbol{\alpha}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ in that right session except with negligible probability. Fix any right session, and assume that that right session is accepting and $\mathcal{A}$ does not cheat in it. Then, from the definition of cheating, that right session has a good row of $\mathsf{sExtCom}$. Let $\{u'_j = (s'_j, d'_j, e'_j)\}_{j\in[10n]}$ be the values that are committed to in that good row of $\mathsf{sExtCom}$. Let $\boldsymbol{s}^{\mathrm{sExt}} = (s_1^{\mathrm{sExt}}, \ldots, s_{10n}^{\mathrm{sExt}})$ be the shares that are derived from $\boldsymbol{u}' = (u'_1, \ldots, u'_{10n})$ as in the definition of the cheating. From the definition of cheating and the robust concurrent extraction lemma, we have the following in that session except with negligible probability.

1. For every $j \in [10n]$, if $s_j^{\mathrm{sExt}} \neq \bot$, it holds $s_j^{\mathrm{sExt}} = s_j^{\mathrm{CEC}} = \alpha_j$.

   (When $s_j^{\mathrm{sExt}} \neq \bot$, the j-th CECom commitment in the row of CECom is valid and has a unique committed value except with negligible probability; therefore, from the robust concurrent extraction lemma, $\alpha_j = s_j^{\mathrm{CEC}}$ holds except with negligible probability.)

2. $\left|\left\{j \text{ s.t. } s_j^{\mathrm{sExt}} = \bot\right\}\right| < n \wedge \left\{j \text{ s.t. } s_j^{\mathrm{sExt}} = \bot\right\} \cap \Gamma = \emptyset$.

3. $\boldsymbol{s}^{\mathrm{sExt}}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $w_j = s_j^{\mathrm{sExt}}$ for every $j \in \Gamma$ or 0.2-far from any such valid codeword.

Hence, from Lemma 6.1 in Section 6.3.1, we have $\mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}}) = \mathsf{Value}_\Gamma(\boldsymbol{\alpha}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{sExt}})$ except with negligible probability. Therefore, if $\mathcal{A}_{\mathrm{cca}}$ does not cheat in an accepting right session, we have $\mathsf{Value}_\Gamma(\boldsymbol{\alpha}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ in that right session except with negligible probability.

Next, we observe that in $G^b_{h:1}(n, z)$, $\mathcal{A}_{\mathrm{cca}}$ cheats in a right session with at most negligible probability. This follows immediately from Subclaim 6.4 (which says that $\mathcal{A}_{\mathrm{cca}}$ cheats in a right session with negligible probability in $H^b_h(n, z)$) and Claim 6.7 (which says that the view of $\mathcal{A}_{\mathrm{cca}}$ in $G^b_{h:1}(n, z)$ is statistically indistinguishable from that in $H^b_h(n, z)$).

From what are observed in the above two paragraphs, it follows that we have $\mathsf{Value}_\Gamma(\boldsymbol{\alpha}) = \mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ in each accepting right session except with negligible probability. □

Proof of Claim 6.9. Recall that $G^b_{h:3}(n, z)$ differs from $G^b_{h:2}(n, z)$ in that the execution of $\mathrm{REAL}^{\mathcal{A}_{\mathrm{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$ (i.e., the robust-concurrent attack between $\mathcal{A}_{\mathrm{robust}}$ and $\mathcal{E}$) is replaced with an interaction between party B of Π and the robust simulator $\mathcal{S}$ of the robust concurrent extraction lemma. Hence, this claim follows immediately from the robust concurrent extraction lemma. (Notice that the round complexity of Π, denoted by R_Π, is $O(R_{\mathsf{CCA}^{1:1}}) = O(\log n)$ and thus the parameter ℓ of CECom satisfies $\ell = \omega(R_\Pi \log n)$.) □

Proof of Claim 6.10. We prove this claim by using the hiding property of $\mathsf{sExtCom}$. Roughly speaking, since $G^b_{k-1:3}(n, z)$ and $G^b_{k:3}(n, z)$ differ only in the shares that are committed to in the row of $\mathsf{sExtCom}$ that $\mathcal{S}$ receives in Π, and $G^b_{k-1:3}(n, z)$ and $G^b_{k:3}(n, z)$ run in polynomial time while $\mathcal{S}$ is receiving the row of $\mathsf{sExtCom}$, the indistinguishability follows directly from the hiding property of $\mathsf{sExtCom}$.

Formally, assume for contradiction that for infinitely many n, there exists $z \in \{0,1\}^*$ such that $\mathsf{G}^b_{k-1:3}(n,z)$ and $\mathsf{G}^b_{k:3}(n,z)$ are distinguishable with advantage $1/\mathsf{poly}(n)$. Since $G^b_{k-1:3}(n,z)$ and $G^b_{k:3}(n,z)$ proceed identically until B starts sending the row of $\mathsf{sExtCom}$ to $\mathcal{S}$, there exists a prefix ρ of a transcript of $G^b_{k-1:3}(n,z)$ up until the row of $\mathsf{sExtCom}$ (exclusive) such that under the condition that ρ is a prefix of the transcript, $\mathsf{G}^b_{k-1:3}(n,z)$ and $\mathsf{G}^b_{k:3}(n,z)$ are distinguishable with advantage $1/\mathsf{poly}(n)$. Note that ρ contains the entire transcript of the $\mathsf{CCACom}^{1:1}$ commitment that $\mathcal{S}$ sends to B, and thus ρ uniquely determines the committed value Γ of this $\mathsf{CCACom}^{1:1}$ commitment. We then consider the following PPT adversary $\mathcal{B}$ against the hiding property of $\mathsf{sExtCom}$.

- Taking ρ and Γ as auxiliary inputs, $\mathcal{B}$ internally invokes $\mathcal{S}$ and emulates $G^b_{k-1:3}(n,z)$ from ρ by receiving either commitments to $\{u_j\}_{j\notin\Gamma}$ or commitments to $\{0^{|u_j|}\}_{j\notin\Gamma}$ from the external committer and then forwarding them to $\mathcal{S}$. Finally, $\mathcal{B}$ outputs whatever $\mathcal{S}$ outputs.

Since $\mathcal{B}$ perfectly emulates either $G^b_{k-1:3}(n,z)$ or $G^b_{k:3}(n,z)$ depending on the commitments it receives, our assumption implies that $\mathcal{B}$ distinguishes commitments to $\{u_j\}_{j\notin\Gamma}$ and commitments to $\{0^{|u_j|}\}_{j\notin\Gamma}$ with advantage $1/\mathsf{poly}(n)$. Thus, we reach a contradiction. □

As noted before, Claim 6.4 follows immediately from Claims 6.7 – 6.10. This concludes the proof of Claim 6.4. □

6.5.1.2 Proof of Subclaim 6.4

We now prove Subclaim 6.4, which says that $\mathcal{A}_{\mathrm{cca}}$ cheats in a right session in $H^b_h(n,z)$ with at most negligible probability.

Proof of Subclaim 6.4. First, we introduce notations. For any $q \in \mathbb{N}$, we say that a right session has *end-index* q if this session is the q-th right session that $\mathcal{A}_{\mathrm{cca}}$ completes. Similarly, we say that a right session has *start-index* q if this session is the q-th right session that $\mathcal{A}_{\mathrm{cca}}$ starts. Note that the end-index of a session is undefined until the session completes, whereas the start-index is defined when the session starts. Jumping ahead, in the proof, we assume for contradiction that there exists an end-index q_{end} such that $\mathcal{A}_{\mathrm{cca}}$ cheats in the session having end-index q_{end}. Then, since we do not know which session has the end-index q_{end} until the session completes, we guess a start-index q_{start} such that the session having the start-index q_{start} has the end-index q_{end}.

We argue that $\mathcal{A}_{\mathrm{cca}}$ cannot cheat in any right session because of the hiding property of $\mathsf{CCACom}^{1:1}$. However, there are two problems.

- Since $\mathcal{A}_{\mathrm{cca}}$ interacts with the committed-value oracle $\mathcal{O}$, which runs in super-polynomial-time, we cannot directly use the *computational* hiding property of $\mathsf{CCACom}^{1:1}$. We overcome this problem by considering a hybrid experiment in which $\mathcal{O}$ is emulated in polynomial time.

- $\mathcal{A}_{\mathrm{cca}}$ may cheat in a right session by using the messages that it receives in the left session, in which the left committer cheats. We overcome this problem by using one-one CCA-security of $\mathsf{CCACom}^{1:1}$ instead of its hiding property. Since

the left session can be emulated in polynomial time given the committed value Γ of the $\mathsf{CCACom}^{1:1}$ commitment in the left session, one-one CCA-security of $\mathsf{CCACom}^{1:1}$ guarantees that the $\mathsf{CCACom}^{1:1}$ commitment in each right session is hiding even when the left committer cheats.

When simulating O in polynomial time, we use the concurrent extractability of CECom for obtaining the shares that are committed to in the row of CECom. Since we want to use the one-one CCA security of $\mathsf{CCACom}^{1:1}$, we use the robust concurrent extraction lemma so that we can use the one-one CCA security of $\mathsf{CCACom}^{1:1}$ even in the presence of the concurrent extraction from CECom.

Formally, assume for contradiction that there exists a right session in which $\mathcal{A}_{\mathrm{cca}}$ cheats with non-negligible probability. Then, there exists an end-index q_{end} such that (i) $\mathcal{A}_{\mathrm{cca}}$ cheats with at most negligible probability in any right session having an end-index less than q_{end}, but (ii) $\mathcal{A}_{\mathrm{cca}}$ cheats with non-negligible probability in the session having end-index q_{end}.

To reach a contradiction, we consider the following hybrid experiments $F^b_{h:1}(n,z),\ldots,F^b_{h:4}(n,z)$.

- Hybrid $F^b_{h:1}(n,z)$ is the same as $H^b_h(n,z)$ except that $F^b_{h:1}(n,z)$ halts immediately after $\mathcal{A}_{\mathrm{cca}}$ completes the session having end-index q_{end} (and immediately before O returns the committed value of this session to $\mathcal{A}_{\mathrm{cca}}$). Note that in $F^b_{h:1}(n,z)$, O returns the committed values to $\mathcal{A}_{\mathrm{cca}}$ only in the right sessions having the end-index less than q_{end}, and $\mathcal{A}_{\mathrm{cca}}$ cheats in those sessions only with negligible probability.

- Hybrid $F^b_{h:2}(n,z)$ is the same as $F^b_{h:1}(n,z)$ except that at the end of each right session, the oracle O returns $\mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ to $\mathcal{A}_{\mathrm{cca}}$ rather than $\mathsf{Value}_\Gamma(\boldsymbol{s})$ as the committed value of this session, where $\boldsymbol{s}=(s_1,\ldots,s_{10n})$ is the shares that are committed to in the row of Com in Stage 2, $\boldsymbol{s}^{\mathrm{CEC}}=(s^{\mathrm{CEC}}_1,\ldots,s^{\mathrm{CEC}}_{10n})$ is the shares that are committed to in the row of CECom in Stage 3, and Γ is the subset that is committed to in the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1.

- Hybrid $F^b_{h:3}(n,z)$ is the same as $F^b_{h:2}(n,z)$ except for syntactical differences: Roughly speaking, $F^b_{h:3}(n,z)$ is an experiment in which $F^b_{h:2}(n,z)$ is executed in such a way that we can use the robust concurrent extraction lemma (Lemma 2.1) later. Formally, $F^b_{h:3}(n,z)$ is defined as follows. Recall that in the setting of the robust concurrent extraction lemma, an adversary, $\mathcal{A}_{\mathrm{robust}}$, launches the robust-concurrent attack by interacting with the online extractor $\mathcal{E}$; specifically, $\mathcal{A}_{\mathrm{robust}}$ interacts with $\mathcal{E}$ as a party A of an arbitrary two-party protocol $\Pi=\langle B,A\rangle$ while interacting with $\mathcal{E}$ as the committers of CECom concurrently and obtaining a value from $\mathcal{E}$ at the end of each session of CECom (where the values that are returned from $\mathcal{E}$ are supposed to be the committed values of the CECom sessions). Then, consider the following Π and $\mathcal{A}_{\mathrm{robust}}$ (see also Figure 6.10).

  $\Pi=\langle B,A\rangle$**:** Parties A and B do the following two interactions concurrently. (The schedule is controlled by A.)

  Interaction 1. A gives a $\mathsf{CCACom}^{1:1}$ commitment to B, where the tag is chosen by A. Then, B extracts the committed value of this $\mathsf{CCACom}^{1:1}$

commitment, denoted by Γ_{left}, by brute force and sends it back to A. (If the $\mathsf{CCACom}^{1:1}$ commitment is invalid, Γ_{left} is set to be a random subset, and if the $\mathsf{CCACom}^{1:1}$ commitment has more than one committed value, B outputs fail and terminates.)

Interaction 2. First, B commits to a random subset $\Gamma_{\text{right}} \subset [10n]$ of size n using $\mathsf{CCACom}^{1:1}$, where the tag is chosen by A. Next, A sends a transcript $\mathcal{T}$ of Stages 2 and 3 of CCACom (i.e., a row of Com followed by a row of CECom), and then gives η rows of $\mathsf{sExtCom}$ to B, where each row consists of $10n$ parallel $\mathsf{sExtCom}$ commitments. (Recall that η is the number of the rows of $\mathsf{sExtCom}$ in CCACom.) Finally, B decommits the $\mathsf{CCACom}^{1:1}$ commitment to Γ_{right}.

$\mathcal{A}_{\text{robust}}$**:** $\mathcal{A}_{\text{robust}}$ takes non-uniform advice z and internally executes $F^b_{h:2}(n, z)$ as follows. (Recall that the execution of $F^b_{h:2}(n, z)$ involves an interaction with the CCA-security adversary $\mathcal{A}_{\text{cca}}$.)

- A start-index q_{start} is chosen at random at the beginning.
- In the left session, $\mathcal{A}_{\text{robust}}$ receives a $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{A}_{\text{cca}}$ in Stage 1 and forwards it to the online extractor $\mathcal{E}$ (who internally emulates party B of Π). Then, instead of extracting the committed subset Γ_{left} from this $\mathsf{CCACom}^{1:1}$ commitment by brute force, $\mathcal{A}_{\text{robust}}$ obtains Γ_{left} from $\mathcal{E}$. Subsequently, $\mathcal{A}_{\text{robust}}$ emulates the left session for $\mathcal{A}_{\text{cca}}$ honesty by using Γ_{left}.
- In the right session having start-index q_{start}, $\mathcal{A}_{\text{robust}}$ receives a $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{E}$ (who internally emulates party B of Π) and forwards it to $\mathcal{A}_{\text{cca}}$ in Stage 1. Then, $\mathcal{A}_{\text{robust}}$ emulates Stages 2 and 3 for $\mathcal{A}_{\text{cca}}$ honestly and sends the transcript $\mathcal{T}$ of these stages to $\mathcal{E}$. Then, $\mathcal{A}_{\text{robust}}$ receives η rows of $\mathsf{sExtCom}$ from $\mathcal{A}_{\text{cca}}$ in Stage 4 and forwards them to $\mathcal{E}$. Then, $\mathcal{A}_{\text{robust}}$ receives a decommitment for the $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{E}$ and forwards it to $\mathcal{A}_{\text{cca}}$ in Stage 5. Then, $\mathcal{A}_{\text{robust}}$ emulates Stage 6 for $\mathcal{A}_{\text{cca}}$ honestly.
- In every other right session, $\mathcal{A}_{\text{robust}}$ emulates Stages 1 – 6 honestly except for forwarding the row of CECom in Stage 3 to $\mathcal{E}$ (who internally emulates the receivers of CECom). Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ denote the responses from $\mathcal{E}$ at the end of the row of CECom. Then, at the end of the right session, $\mathcal{A}_{\text{robust}}$ sends $\mathsf{Value}_{\Gamma}(\boldsymbol{\alpha})$ to $\mathcal{A}_{\text{cca}}$ as the committed value of this right session.

The output of $\mathcal{A}_{\text{robust}}$ is that of the internally executed $F^b_{h:2}(n, z)$.

From the robust concurrent extraction lemma, there exists a robust simulator $\mathcal{S}$ such that for the above $\mathcal{A}_{\text{robust}}$, there exists an online extractor $\mathcal{E}$ that satisfies the following.

- For any row of CECom that $\mathcal{A}_{\text{robust}}$ sends to $\mathcal{E}$, let $\boldsymbol{s}^{\text{CEC}} = (s^{\text{CEC}}_1, \ldots, s^{\text{CEC}}_{10n})$ be the shares that are committed to in this row of CECom and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ be the responses from $\mathcal{E}$ at the end of this row. Then, for every $j \in [10n]$, if the j-th CECom commitment in this row is valid and

its committed value is uniquely determined, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ satisfies $\alpha_j = s_j^{\mathrm{CEC}}$.

– $\mathcal{S}$ can simulate the robust-concurrent attack between $\mathcal{A}_{\mathrm{robust}}$ and $\mathcal{E}$.

Hybrid $F^b_{h:3}(n, z)$ is the experiment $\mathrm{REAL}^{\mathcal{A}_{\mathrm{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$ of the robust concurrent extraction lemma. The output of $F^b_{h:3}(n, z)$ is that of $\mathcal{A}_{\mathrm{robust}}$ in $\mathrm{REAL}^{\mathcal{A}_{\mathrm{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$.

In what follows, we say that $\mathcal{A}_{\mathrm{robust}}$ *cheats* in $F^b_{h:3}(n)$ if in the execution of $F^b_{h:2}(n, z)$ that is emulated by $\mathcal{A}_{\mathrm{robust}}$ in $F^b_{h:3}(n)$, $\mathcal{A}_{\mathrm{cca}}$ cheats in the right session having start-index q_{start}. We remark that, since $\mathcal{A}_{\mathrm{robust}}$ sends the transcript $\mathcal{T}$ of Stages 2 and 3 to $\mathcal{E}$ in Π, we can see whether $\mathcal{A}_{\mathrm{robust}}$ cheats in $F^b_{h:3}(n)$ or not by examining the transcript of Π between $\mathcal{A}_{\mathrm{robust}}$ and $\mathcal{E}$ (specifically, by extracting the committed values from each row of $\mathsf{sExtCom}$ by brute force and then checking whether those committed values satisfy the badness condition in Definition 6.5 w.r.t. Stages 2 and 3 of CCACom that appear in $\mathcal{T}$).

- Hybrid $F^b_{h:4}(n, z)$ differs from $F^b_{h:3}(n, z)$ in that the execution of $\mathrm{REAL}^{\mathcal{A}_{\mathrm{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$ (i.e., the robust-concurrent attack between $\mathcal{A}_{\mathrm{robust}}$ and $\mathcal{E}$) is replaced with an interaction between party B of Π and the robust simulator $\mathcal{S}$ of the robust concurrent extraction lemma (see Figure 6.11). The output of $F^b_{h:4}(n, z)$ is that of $\mathcal{A}_{\mathrm{robust}}$ that is simulated by $\mathcal{S}$.

  In what follows, we say that $\mathcal{S}$ *cheats* in $F^b_{h:4}(n, z)$ if $\mathcal{A}_{\mathrm{robust}}$ cheats in the view that is simulated by $\mathcal{S}$.

First, we notice that in $F^b_{h:1}(n, z)$, $\mathcal{A}_{\mathrm{cca}}$ cheats with at most negligible probability in any right session having an end-index less than q_{end}, and $\mathcal{A}_{\mathrm{cca}}$ cheats with non-negligible probability in the session having end-index q_{end}. This is because $F^b_{h:1}(n, z)$ proceeds identically with $H^b_h(n, z)$ until the end of the right session having end-index q_{end}.

Next, we observe that in $F^b_{h:2}(n, z)$, $\mathcal{A}_{\mathrm{cca}}$ cheats with non-negligible probability in the session having end-index q_{end}. Recall that $F^b_{h:2}(n, z)$ differs from $F^b_{h:1}(n, z)$ in that at the end of each right session having an end-index less than q_{end}, the oracle O computes the committed value of the session by $\mathsf{Value}_\Gamma(s^{\mathrm{CEC}})$ rather than by $\mathsf{Value}_\Gamma(s)$. Then, since in $F^b_{h:1}(n, z)$ $\mathcal{A}_{\mathrm{cca}}$ cheats with at most negligible probability in any right session having an end-index less than q_{end}, we can show that $\mathsf{Value}_\Gamma(s^{\mathrm{CEC}}) = \mathsf{Value}_\Gamma(s)$ holds in any such right session except with negligible probability by using the same argument as in the proof of Subclaim 6.3. Hence, the view of $\mathcal{A}_{\mathrm{cca}}$ in $F^b_{h:2}(n, z)$ is statistically indistinguishable from that in $F^b_{h:1}(n, z)$, so $\mathcal{A}_{\mathrm{cca}}$ cheats with non-negligible probability in the session having end-index q_{end} in $F^b_{h:2}(n, z)$.

Next, we observe that $\mathcal{A}_{\mathrm{robust}}$ cheats in $F^b_{h:3}(n, z)$ with non-negligible probability. From the construction of $F^b_{h:3}(n, z)$, an execution of $F^b_{h:2}(n, z)$ is perfectly emulated in $F^b_{h:3}(n, z)$ as long as we have $\mathsf{Value}_\Gamma(\boldsymbol{\alpha}) = \mathsf{Value}_\Gamma(s^{\mathrm{CEC}})$ in each accepting right session that has an end-index less than q_{end}. Then, since in $F^b_{h:2}(n, z)$ $\mathcal{A}_{\mathrm{cca}}$ cheats with at most negligible probability in any right session having an end-index less than q_{end}, we can show that $\mathsf{Value}_\Gamma(\boldsymbol{\alpha}) = \mathsf{Value}_\Gamma(s^{\mathrm{CEC}})$ holds in any such right session except with negligible probability by using the same argument as in the proof of Claim 6.8. Hence, in the execution of $F^b_{h:2}(n, z)$ that is emulated in $F^b_{h:3}(n, z)$, $\mathcal{A}_{\mathrm{cca}}$ cheats with non-negligible

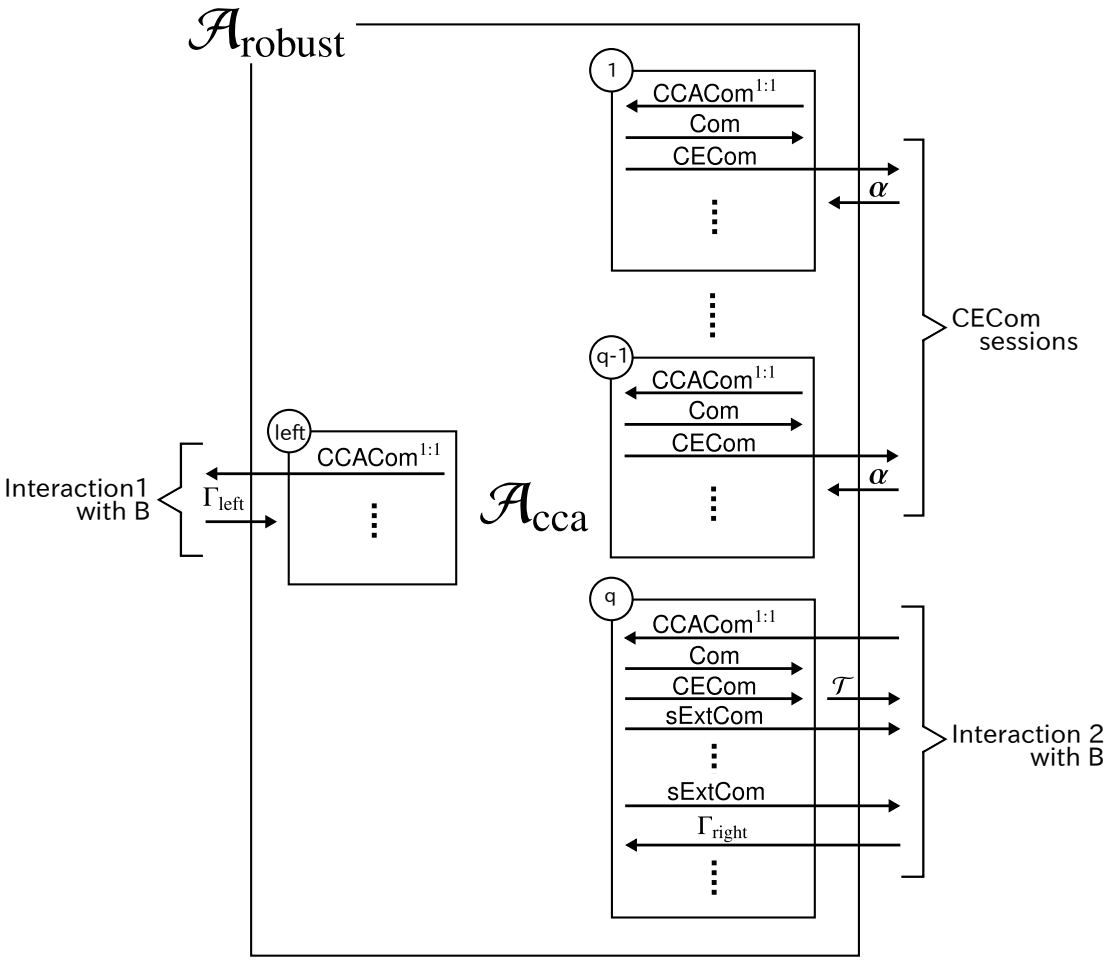

Figure 6.10: Adversary $\mathcal{A}_{\text{robust}}$ in Hybrid $F^b_{h:3}(n,z)$. For simplicity, the right sessions are illustrated as if they are executed sequentially.

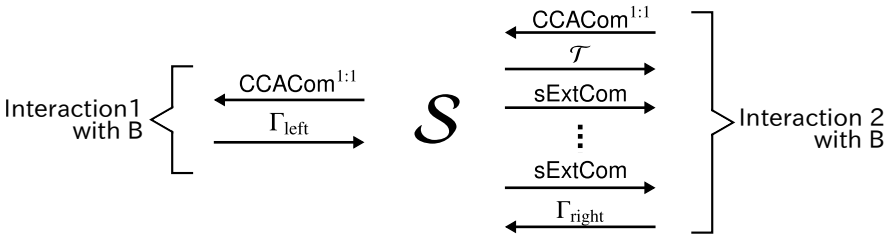

Figure 6.11: Simulator $\mathcal{S}$ in Hybrid $F^b_{h:4}(n,z)$.

probability in the session having end-index q_{end}. Now, since the number of the right sessions is polynomially bounded, we conclude that in the execution of $F^b_{h:2}(n,z)$ that is emulated in $F^b_{h:3}(n,z)$, $\mathcal{A}_{\mathrm{cca}}$ cheats with non-negligible probability in the session having start-index q_{start}.

Next, we observe that $\mathcal{S}$ cheats in $F^b_{h:4}(n,z)$ with non-negligible probability. This follows from the robust concurrent extraction lemma, which guarantees that $\mathcal{A}_{\mathrm{robust}}$'s view is statistically simulated in $F^b_{h:4}(n,z)$. (Notice that the round complexity R_Π of Π is $O(R_{\mathsf{CCA}^{1:1}}) = O(\log n)$ and thus the parameter ℓ of CECom satisfies $\ell = \omega(R_\Pi \log n)$.)

We then derive a contradiction by showing that we can break the one-one CCA security of $\mathsf{CCACom}^{1:1}$ using $F^b_{h:4}(n,z)$.

For a warm up, we first consider the following super-polynomial-time adversary $\mathcal{M}'$ against the one-one CCA security of $\mathsf{CCACom}^{1:1}$ (see also Figure 6.12).

- Externally, $\mathcal{M}'$ sends random subsets $\Gamma_0, \Gamma_1 \subset [10n]$ to a committer of $\mathsf{CCACom}^{1:1}$ and receives a $\mathsf{CCACom}^{1:1}$ commitment from it (the committed value is either Γ_0 or Γ_1). Concurrently, $\mathcal{M}'$ also interacts with the committed-value oracle of $\mathsf{CCACom}^{1:1}$ in a single session.

  Internally, $\mathcal{M}'$ invokes $\mathcal{S}$ and emulates $F^b_{h:4}(n,z)$ for $\mathcal{S}$ honestly except for the following.

  - When sending a $\mathsf{CCACom}^{1:1}$ commitment to $\mathcal{S}$ as the commitment from B in Π, $\mathcal{M}'$ obtains a $\mathsf{CCACom}^{1:1}$ commitment from the external committer and forwards it to $\mathcal{S}$.
  - When $\mathcal{S}$ starts sending a $\mathsf{CCACom}^{1:1}$ commitment to B in Π, $\mathcal{M}'$ forwards it to external O, and then, instead of extracting its committed value Γ_{left} by brute force, $\mathcal{M}'$ obtains Γ_{left} from O.
  - When $\mathcal{S}$ starts sending η rows of $\mathsf{sExtCom}$ to B in Π, $\mathcal{M}'$ extracts the committed values of an arbitrarily chosen row by brute force. $\mathcal{M}'$ then stops emulating $F^b_{h:4}(n,z)$.

  Let $\{u_j = (s_j, d_j, e_j)\}_{j\in[10n]}$ be the values that are extracted from the arbitrarily chosen row of $\mathsf{sExtCom}$, and $\mathcal{T}$ be the message that $\mathcal{S}$ sends to B in Π as the transcript of Stages 2 and 3 of CCACom. Let $\boldsymbol{s}^{\mathrm{sExt}} = (s_1^{\mathrm{sExt}}, \ldots, s_{10n}^{\mathrm{sExt}})$ be the shares that are derived from $\boldsymbol{u} = (u_1, \ldots, u_{10n})$ and $\mathcal{T}$ as in the definition of the cheating (Definition 6.5). Then, $\mathcal{M}'$ outputs 1 if and only if either of the following holds.

  1. $\left|\left\{j \in [10n] \text{ s.t. } s_j^{\mathrm{sExt}} = \bot\right\}\right| \geq n \wedge \left\{j \in [10n] \text{ s.t. } s_j^{\mathrm{sExt}} = \bot\right\} \cap \Gamma_1 = \emptyset$.
  2. $\boldsymbol{s}^{\mathrm{sExt}}$ is 0.8-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $s_j^{\mathrm{sExt}} = w_j$ for every $j \in \Gamma_1$, but $\boldsymbol{s}^{\mathrm{sExt}}$ is 0.1-far from $\boldsymbol{w}$.

When $\mathcal{M}'$ receives a commitment to Γ_0, $\mathcal{M}'$ outputs 1 only with negligible probability; this is because when $\mathcal{M}'$ receives a commitment to Γ_0, the internal $\mathcal{S}$ receives no information about Γ_1, and thus, the probability that either of the following holds is negligible.

1. $\left|\left\{j \in [10n] \text{ s.t. } s_j^{\mathrm{sExt}} = \bot\right\}\right| \geq n$ but $\left\{j \in [10n] \text{ s.t. } s_j^{\mathrm{sExt}} = \bot\right\} \cap \Gamma_1 = \emptyset$.

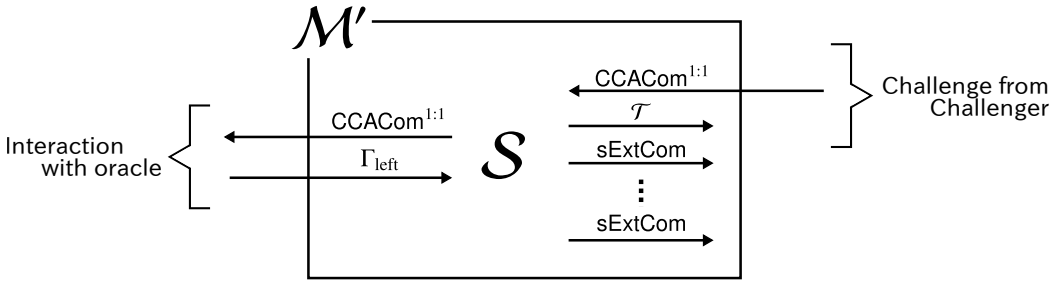

Figure 6.12: Adversary $\mathcal{M}'$ against the one-one CCA security of CCACom$^{1:1}$.

2. $\boldsymbol{s}^{\text{sExt}}$ is 0.1-far from a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ but we have $s_j^{\text{sExt}} = w_j$ for every $j \in \Gamma_1$.

On the other hand, when $\mathcal{M}'$ receives a commitment to Γ_1, the internal $\mathcal{S}$ cheats in the emulated execution of $F_{h:4}^b(n, z)$ with non-negligible probability, so all the rows of sExtCom from $\mathcal{S}$ are bad (w.r.t. Stages 2 and 3 of CCACom that appear in $\mathcal{T}$) with non-negligible probability; hence, from the definition of cheating (Definition 6.5), $\mathcal{M}'$ outputs 1 with non-negligible probability. Thus, $\mathcal{M}'$ distinguishes a commitment to Γ_0 and a commitment to Γ_1 with non-negligible advantage.

We then consider an adversary $\mathcal{M}$ that emulates $\mathcal{M}'$ in polynomial time by extracting the committed values of a row of sExtCom by using the extractability of sExtCom. To formally define $\mathcal{M}$, we first define the following machine $\widehat{\mathcal{M}}$.

- Externally, $\widehat{\mathcal{M}}$ sends random subsets $\Gamma_0, \Gamma_1 \subset [10n]$ to an external party, receives a CCACom$^{1:1}$ commitment from a committer of CCACom$^{1:1}$ (the committed value is either Γ_0 or Γ_1), sends a transcript $\mathcal{T}$ of Stages 2 and 3 of CCACom to an external party, and then gives a row of sExtCom to a receiver of sExtCom. Concurrently, $\widehat{\mathcal{M}}$ also interacts with the committed-value oracle of CCACom$^{1:1}$ in a single session.

  Internally, $\widehat{\mathcal{M}}$ invokes $\mathcal{S}$ and emulates $F_{h:4}^b(n, z)$ for $\mathcal{S}$ honestly except for the following.

  - When sending a CCACom$^{1:1}$ commitment to $\mathcal{S}$ as the commitment from B in Π, $\widehat{\mathcal{M}}$ obtains a CCACom$^{1:1}$ commitment from the external committer and forwards it to $\mathcal{S}$.
  - When $\mathcal{S}$ starts sending a CCACom$^{1:1}$ commitment to B in Π, $\widehat{\mathcal{M}}$ forwards it to external $\mathcal{O}$, and then, instead of extracting its committed value Γ_{left} by brute force, $\widehat{\mathcal{M}}$ obtains Γ_{left} from $\mathcal{O}$.
  - After receiving a transcript $\mathcal{T}$ of Stages 2 and 3 of CCACom from $\mathcal{S}$, $\widehat{\mathcal{M}}$ forwards it to the external party.
  - When $\mathcal{S}$ starts sending η rows of sExtCom to B in Π, $\widehat{\mathcal{M}}$ forwards a randomly chosen row among them to the external receiver of sExtCom. If the randomly chosen row of sExtCom "interleaves" with any messages of the CCACom$^{1:1}$ commitment that are being forwarded to $\mathcal{O}$ (namely, if $\mathcal{S}$ tries

to send/receive a message of that $\mathsf{CCACom}^{1:1}$ commitment while sending that row of $\mathsf{sExtCom}$), $\widehat{\mathcal{M}}$ stops emulating $F^b_{h:4}(n,z)$ immediately and terminates. In other cases, $\widehat{\mathcal{M}}$ stops emulating $F^b_{h:4}(n,z)$ and terminates when the randomly chosen row of $\mathsf{sExtCom}$ completes.

We remark that once $\widehat{\mathcal{M}}$ starts sending a $\mathsf{sExtCom}$ commitment to the external receiver of $\mathsf{sExtCom}$, $\widehat{\mathcal{M}}$ no longer interacts with the oracle $\mathcal{O}$. (Once $\widehat{\mathcal{M}}$ starts sending a $\mathsf{sExtCom}$ commitment, either $\widehat{\mathcal{M}}$ terminates in the middle of $\mathsf{sExtCom}$ (because the internal $\mathcal{S}$ tries to send/receive a message of $\mathsf{CCACom}^{1:1}$) or $\widehat{\mathcal{M}}$ completes the $\mathsf{sExtCom}$ commitment.) Furthermore, since $\eta = R_{\mathsf{CCA}^{1:1}} + 1$ (and thus the number of rows of $\mathsf{sExtCom}$ is bigger than the number of rounds in $\mathsf{CCACom}^{1:1}$), a randomly chosen row of $\mathsf{sExtCom}$ does not interleave with any messages of $\mathsf{CCACom}^{1:1}$ with non-negligible probability; thus, $\widehat{\mathcal{M}}$ completes the $\mathsf{sExtCom}$ commitment with non-negligible probability.

Using $\widehat{\mathcal{M}}$, we define $\mathcal{M}$ as follows.

- Externally, $\mathcal{M}$ sends random subsets $\Gamma_0, \Gamma_1 \subset [10n]$ to a committer of $\mathsf{CCACom}^{1:1}$ and receives a $\mathsf{CCACom}^{1:1}$ commitment from it (the committed value is either Γ_0 or Γ_1). Concurrently, $\mathcal{M}$ also interacts with the committed-value oracle of $\mathsf{CCACom}^{1:1}$ in a single session.

  Internally, $\mathcal{M}$ invokes $\widehat{\mathcal{M}}$ and lets it interact with the external committer of $\mathsf{CCACom}^{1:1}$ and the oracle $\mathcal{O}$. When $\widehat{\mathcal{M}}$ starts sending a row of $\mathsf{sExtCom}$, $\mathcal{M}$ invokes the extractor of $\mathsf{sExtCom}$ against $\widehat{\mathcal{M}}$ and obtains (τ, σ), where τ is the view of $\widehat{\mathcal{M}}$ as a committer of $\mathsf{sExtCom}$ and σ is a possible value that $\widehat{\mathcal{M}}$ committed to in τ.

  If the $\mathsf{sExtCom}$ commitment that $\widehat{\mathcal{M}}$ gives in τ is not accepting or the extractor of $\mathsf{sExtCom}$ fails (i.e., the commitment in τ is accepting but $\sigma = \bot$ holds), $\mathcal{M}$ outputs 0. Otherwise, parse σ as $\{u_j = (s_j, d_j, e_j)\}_{j\in[10n]}$, and let $\mathcal{T}$ be the transcript that $\mathcal{M}$ obtained from $\widehat{\mathcal{M}}$ before the row of $\mathsf{sExtCom}$. Let $\boldsymbol{s}^{\mathsf{sExt}} = (s_1^{\mathsf{sExt}}, \ldots, s_{10n}^{\mathsf{sExt}})$ be the shares that are derived from $\boldsymbol{u} = (u_1, \ldots, u_{10n})$ and $\mathcal{T}$ as in the definition of the cheating (Definition 6.5). Then, $\mathcal{M}$ outputs 1 if and only if either of the following holds.

  1. $\left|\left\{j \in [10n] \text{ s.t. } s_j^{\mathsf{sExt}} = \bot\right\}\right| \geq n \wedge \left\{j \in [10n] \text{ s.t. } s_j^{\mathsf{sExt}} = \bot\right\} \cap \Gamma_1 = \emptyset$.
  2. $\boldsymbol{s}^{\mathsf{sExt}}$ is 0.8-close to a valid codeword $\boldsymbol{w} = (w_1, \ldots, w_{10n})$ that satisfies $s_j^{\mathsf{sExt}} = w_j$ for every $j \in \Gamma_1$, but $\boldsymbol{s}^{\mathsf{sExt}}$ is 0.1-far from $\boldsymbol{w}$.

Recall that, as observed above, $\widehat{\mathcal{M}}$ gives an accepting $\mathsf{sExtCom}$ commitment with non-negligible probability. Furthermore, the extractor of $\mathsf{sExtCom}$ fails only with negligible probability, and no over-extraction occur during the extraction. Hence, from exactly the same argument as in the analysis of $\mathcal{M}'$ above, $\mathcal{M}$ distinguishes a commitment to Γ_0 and a commitment to Γ_1 with non-negligible advantage. Since $\mathcal{M}$ runs in polynomial time, this is a contradiction. □

6.5.1.3 Proof of Subclaim 6.2

We now prove Subclaim 6.2, which says that $H^b_k(n,z)$ outputs fail with at most negligible probability. Recall that $H^b_k(n,z)$ outputs fail when the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1 of the left session has more than one committed value.

Proof of Subclaim 6.2. Since $H^b_k(n,z)$ outputs fail only if the commitment in Stage 1 has more than one committed value in the left session, we prove this claim by using the binding property of $\mathsf{CCACom}^{1:1}$. A problem is that $\mathcal{A}_{\mathrm{cca}}$ interacts with the committed-value oracle $\mathcal{O}$, which runs in super-polynomial time; because of the super-polynomial-time power of $\mathcal{O}$, the claim does not follow directly from the strong computational binding property of $\mathsf{CCACom}^{1:1}$. We overcome this problem by, again, emulating $\mathcal{O}$ in polynomial time using the robust concurrent extraction lemma on CECom. The proof is similar to that of Claim 6.4.

Formally, assume for contradiction that $H^b_k(n,z)$ outputs fail with non-negligible probability. Then, the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1 of the left session has more than one committed value with non-negligible probability.

We consider the following hybrid experiments.

Hybrid $E^b_{k:1}(n,z)$: Hybrid $E^b_{k:1}(n,z)$ is the same as $H^b_k(n,z)$ except for the following.

- At the end of each right session, the oracle $\mathcal{O}$ returns $\mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ to $\mathcal{A}_{\mathrm{cca}}$ rather than $\mathsf{Value}_\Gamma(\boldsymbol{s})$ as the committed value of this session, where $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ is the shares that are committed to in the row of Com in Stage 2, $\boldsymbol{s}^{\mathrm{CEC}} = (s^{\mathrm{CEC}}_1, \ldots, s^{\mathrm{CEC}}_{10n})$ is the shares that are committed to in the row of CECom in Stage 3, and Γ is the subset that is committed to in the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1.
- The experiment is terminated at the end of Stage 1 of the left session.

Hybrid $E^b_{k:2}(n,z)$: Hybrid $E^b_{k:2}(n,z)$ is the same as $E^b_{k:1}(n,z)$ except for syntactical differences: Roughly speaking, $E^b_{k:2}(n,z)$ is an experiment in which $E^b_{k:1}(n,z)$ is executed in such a way that we can use the robust concurrent extraction lemma later (Lemma 2.1). Formally, $E^b_{k:2}(n,z)$ is defined as follows. Recall that in the setting of the robust concurrent extraction lemma, an adversary, $\mathcal{A}_{\mathrm{robust}}$, launches the robust-concurrent attack by interacting with the online extractor $\mathcal{E}$; specifically, $\mathcal{A}_{\mathrm{robust}}$ interacts with $\mathcal{E}$ as a party A of an arbitrary two-party protocol $\Pi = \langle B, A\rangle$ while interacting with $\mathcal{E}$ as the committers of CECom concurrently and obtaining a value from $\mathcal{E}$ at the end of each session of CECom (where the values that are returned from $\mathcal{E}$ are supposed to be the committed values of the CECom sessions). Then, consider the following Π and $\mathcal{A}_{\mathrm{robust}}$.

$\Pi = \langle B, A\rangle$**:** Party A gives a $\mathsf{CCACom}^{1:1}$ commitment to party B, where the tag in the $\mathsf{CCACom}^{1:1}$ commitment is chosen by A.

$\mathcal{A}_{\mathrm{robust}}$**:** $\mathcal{A}_{\mathrm{robust}}$ takes non-uniform advice z and internally executes $E^b_{k:1}(n,z)$ with the following changes. (Recall that the execution of $E^b_{k:1}(n,z)$ involves an interaction with the CCA-security adversary $\mathcal{A}_{\mathrm{cca}}$.)

- In Stage 1 of the left session, $\mathcal{A}_{\text{robust}}$ forwards the $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{A}_{\text{cca}}$ to the online extractor $\mathcal{E}$ (who internally emulates party B of Π).
- In Stage 3 of each right session, $\mathcal{A}_{\text{robust}}$ receives a row of CECom commitments from $\mathcal{A}_{\text{cca}}$ and forwards it to $\mathcal{E}$ (who internally emulates the receivers of CECom). Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ denote the responses from $\mathcal{E}$ at the end of the row of the CECom commitments.
- At the end of each right session, $\mathcal{A}_{\text{robust}}$ sends $\mathsf{Value}_{\Gamma}(\boldsymbol{\alpha})$ to $\mathcal{A}_{\text{cca}}$ as the committed value of this right session.

From the robust concurrent extraction lemma, there exists a robust simulator $\mathcal{S}$ such that for the above $\mathcal{A}_{\text{robust}}$, there exists an online extractor $\mathcal{E}$ that satisfies the following.

- For any row of CECom that $\mathcal{A}_{\text{robust}}$ sends to $\mathcal{E}$, let $\boldsymbol{s}^{\text{CEC}} = (s_1^{\text{CEC}}, \ldots, s_{10n}^{\text{CEC}})$ be the shares that are committed to in this row of CECom and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ be the responses from $\mathcal{E}$ at the end of this row. Then, for every $j \in [10n]$, if the j-th CECom commitment in this row is valid and its committed value is uniquely determined, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ satisfies $\alpha_j = s_j^{\text{CEC}}$.
- $\mathcal{S}$ can simulate the robust-concurrent attack between $\mathcal{A}_{\text{robust}}$ and $\mathcal{E}$.

Hybrid $E^b_{k:2}(n, z)$ is the experiment $\text{REAL}^{\mathcal{A}_{\text{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$ of the robust concurrent extraction lemma.

Hybrid $E^b_{k:3}(n, z)$: Hybrid $E^b_{k:3}(n, z)$ differs from $E^b_{k:2}(n, z)$ in that the execution of $\text{REAL}^{\mathcal{A}_{\text{robust}}}_{\mathcal{E},\Pi}(n, \bot, z)$ (i.e., the robust-concurrent attack by $\mathcal{A}_{\text{robust}}$ against $\mathcal{E}$) is replaced with an interaction between party B of Π and the robust simulator $\mathcal{S}$ of the robust concurrent extraction lemma.

First, we notice that in $E^b_{k:1}(n, z)$, the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1 of the left session has more than one committed value with non-negligible probability. This is because from the same argument as in the proof of Claim 6.7, we can show that the view of $\mathcal{A}_{\text{cca}}$ in $E^b_{k:1}(n, z)$ is statistically close to that in $H^b_k(n, z)$.

Next, we notice that in $E^b_{k:2}(n, z)$, the $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{A}_{\text{robust}}$ to $\mathcal{E}$ has more than one committed value with non-negligible probability. This is because from the same argument as in the proof of Claim 6.8, we can show that an execution of $E^b_{k:1}(n, z)$ is statistically simulated in $E^b_{k:2}(n, z)$.

Next, we notice that in $E^b_{k:3}(n, z)$, the $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{S}$ to B has more than one committed value with non-negligible probability. This is because from the robust concurrent extraction lemma, we can show that the $\mathsf{CCACom}^{1:1}$ commitment between $\mathcal{S}$ and B in $E^b_{k:3}(n, z)$ is statistically close to that between $\mathcal{A}_{\text{robust}}$ and $\mathcal{E}$ in $E^b_{k:2}(n, z)$.

Now, since $E^b_{k:3}(n, z)$ runs in polynomial time and $\mathcal{S}$ interacts with an honest receiver of $\mathsf{CCACom}^{1:1}$ in it, we reach a contradiction to the strong computational binding property of $\mathsf{CCACom}^{1:1}$. This concludes the proof of Subclaim 6.2. □

6.5.1.4 Proofs of Claims 6.5 and 6.6

Claims 6.5 and 6.6 can be proven very similarly to Claim 6.4. For example, consider the case of Claim 6.5, which says that the output of $H^b_\eta(n,z)$ and that of $H^b_{\eta+1}(n,z)$ are computationally indistinguishable. Since $H^b_\eta(n,z)$ and $H^b_{\eta+1}(n,z)$ differ only in the committed values of a row of CECom, we can prove Claim 6.5 by modifying the proof of Claim 6.4 accordingly. (Recall that in the proof of Claim 6.4, our goal is to show the indistinguishability between the outputs of two hybrids that differ only in the values committed to in a row of sExtCom.) The only problem is that the round complexity of CECom is $O(\ell) = \widetilde{O}(\log^2 n)$ (whereas the round complexity of sExtCom is $O(1)$), and thus we cannot use the robust concurrent extraction lemma in the same way as in the proof of Claim 6.4. However, since a CECom commitment can be decomposed into n ExtCom commitments, we can easily solve this problem by designing a sequence of sub-hybrids such that each neighboring sub-hybrids differ in the values that are committed to in a row of ExtCom, which has only $O(1)$ rounds.

Below, we give more details about the proofs of Claims 6.5 and 6.6, which can be skipped with little loss of understanding.

Proof sketch of Claim 6.5. We consider the following sub-hybrids $H^b_{\eta:0}(n,z),\ldots,H^b_{\eta:k}(n,z)$. Recall that a CECom commitment consists of n ExtCom commitments (cf. Figure 2.2 in Section 2.3.3).

Sub-hybrid $H^b_{\eta:0}(n,z)$: Sub-hybrid $H^b_{\eta:0}(n,z)$ is the same as $H^b_\eta(n,z)$.

Sub-hybrid $H^b_{\eta:1}(n,z)$ to Sub-hybrid $H^b_{\eta:n}(n,z)$: For $k \in [n]$, Sub-hybrid $H^b_{\eta:k}(n,z)$ is the same as $H^b_{\eta:0}(n,z)$ except that in Stage 3 of the left session, for every $j \notin \Gamma$ the j-th commitment in the row of CECom is computed as follows. Recall that a CECom commitment consist of n ExtCom commitments. Then, the left committer commits to $0^{|s_j|}$ instead of s_j in the first k ExtCom commitments and commits to s_j in the other $(n-k)$ ExtCom commitments.

Notice that $H^b_{\eta:k}(n,z)$ is identical with $H^b_{\eta+1}(n,z)$.

We can prove Claim 6.5 by showing that the output of $H^b_{\eta:k-1}(n,z)$ and that of $H^b_{\eta:k}(n,z)$ are computationally indistinguishable for each $k \in [n]$, and we can prove this indistinguishability similarly to Claim 6.4. In more detail, we can prove this indistinguishability as follows.

1. Design hybrid experiments $G'^b_{h:1}(n,z),\ldots,G'^b_{h:3}(n,z)$ for $h \in \{k-1,k\}$ in the same way as we design $G^b_{h:1}(n,z),\ldots,G^b_{h:3}(n,z)$ in the proof of Claim 6.4, where the differences from $G^b_{h:1}(n,z),\ldots,G^b_{h:3}(n,z)$ are the following.

   - $G'^b_{h:1}(n,z),\ldots,G'^b_{h:3}(n,z)$ are defined by modifying $H^b_{\eta:h}(n,z)$ rather than $H^b_h(n,z)$.
   - In the definition of $G'^b_{h:2}(n,z)$, party B in the two-party protocol Π sends party A a row of ExtCom commitments rather than a row of sExtCom, and $\mathcal{A}_{\text{robust}}$ forwards the ExtCom commitments from $\mathcal{E}$ to the internally emulated $\mathcal{A}_{\text{cca}}$ as the k-th ExtCom commitment of each CECom commitment

in Stage 3 of the left session (rather than forwarding the sExtCom commitments in a row of sExtCom in the left session).

2. Prove, as in the proofs of Claims 6.7 – 6.10, that the outputs of $H^b_{\eta:h}(n,z), G'^b_{h:1}(n,z), \ldots, G'^b_{h:3}(n,z)$ are computationally indistinguishable for each $h \in \{k-1, k\}$ and that the outputs of $G'^b_{k-1:3}(n,z)$ and $G'^b_{k:3}(n,z)$ are computationally indistinguishable. The only difference from the proofs of Claims 6.7 – 6.10 is that we use the hiding property of ExtCom (rather than that of sExtCom) when proving the indistinguishability between the outputs of $G'^b_{k-1:3}(n,z)$ and $G'^b_{k:3}(n,z)$.

   (When proving these indistinguishabilities, it is required to prove that $\mathcal{A}_{\mathrm{cca}}$ does not cheat in $H^b_{\eta:h}(n,z)$, and this can be proven in the same way as in the proof of Subclaim 6.4.)

3. Use a hybrid argument to conclude that the output of $H^b_{\eta:k-1}(n,z)$ and that of $H^b_{\eta:k}(n,z)$ are computationally indistinguishable.

□

Proof sketch of Claim 6.6. We can prove the indistinguishability between the outputs of $H^b_{\eta+1}(n,z)$ and $H^b_{\eta+2}(n,z)$ similarly to Claim 6.4. In more detail, we can prove this indistinguishability as follows.

1. Design hybrid experiments $G''^b_{h:1}(n,z), \ldots, G''^b_{h:3}(n,z)$ for $h \in \{\eta+1, \eta+2\}$ in the same way as we design $G^b_{h:1}(n,z), \ldots, G^b_{h:3}(n,z)$ in the proof of Claim 6.4, where the difference from $G^b_{h:1}(n,z), \ldots, G^b_{h:3}(n,z)$ is the following.

   - In the definition of $G''^b_{h:2}(n,z)$, party B in the two-party protocol Π sends party A a row of Com commitments rather than a row of sExtCom, and $\mathcal{A}_{\mathrm{robust}}$ forwards the Com commitments from $\mathcal{E}$ to the internally emulated $\mathcal{A}_{\mathrm{cca}}$ as the row of Com in Stage 2 of the left session (rather than forwarding the sExtCom commitments in a row of sExtCom in the left session).

2. Prove, as in the proofs of Claims 6.7 – 6.10, that the outputs of $H^b_h(n,z), G''^b_{h:1}(n,z), \ldots, G''^b_{h:3}(n,z)$ are computationally indistinguishable for each $h \in \{\eta+1, \eta+2\}$ and that the outputs of $G''^b_{\eta+1:3}(n,z)$ and $G''^b_{\eta+2:3}(n,z)$ are computationally indistinguishable. The only difference from the proofs of Claims 6.7 – 6.10 is that we use the hiding property of Com (rather than that of sExtCom) when proving the indistinguishability between the outputs of $G''^b_{\eta+1:3}(n,z)$ and $G''^b_{\eta+2:3}(n,z)$.

3. Use a hybrid argument to conclude that the output of $H^b_{\eta+1}(n,z)$ and that of $H^b_{\eta+2}(n,z)$ are computationally indistinguishable.

□

Combining Claims 6.4, 6.5, and 6.6 and Equation (6.14), we obtain Lemma 6.4. This concludes the proof of Lemma 6.4. □

6.5.2 Proof of Robustness

Lemma 6.5. *For any constant $\kappa \in \mathbb{N}$,* CCACom *is κ-robust.*

Like the robustness of previous CCA-secure commitments [CLP10, CLP16, LP12], the robustness of our CCA-secure commitment can be shown by using the techniques in the proof of its CCA security.

Proof of Lemma 6.5. We show that there exists a PPT simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$ and any κ-round PPT ITM B, the following indistinguishability holds.

$$\begin{aligned}&\left\{\mathsf{output}_{B,\mathcal{A}^{\mathcal{O}}}\left[B(1^n,y)\leftrightarrow\mathcal{A}^{\mathcal{O}}(1^n,z)\right]\right\}_{n\in\mathbb{N},y,z\in\{0,1\}^n}\\&\overset{c}{\approx}\left\{\mathsf{output}_{B,\mathcal{S}}\left[B(1^n,y)\leftrightarrow\mathcal{S}(1^n,z)\right]\right\}_{n\in\mathbb{N},y,z\in\{0,1\}^n}\end{aligned}\tag{6.15}$$

First, we consider the following hybrid experiments.

Hybrid $D_0(n,y,z)$: Hybrid $D_0(n,y,z)$ is an experiment in which $\mathcal{A}^{\mathcal{O}}(1^n,x,z)$ interacts with party $B(1^n,y,z)$ as in the definition of robustness, i.e., $\mathcal{A}$ interacts with B while interacting with the committed-value oracle $\mathcal{O}$ in concurrent sessions of CCACom. The output of the experiment is the joint output of B and $\mathcal{A}$.

Hybrid $D_1(n,y,z)$: Hybrid $D_1(n,y,z)$ is the same as $D_0(n,y,z)$ except that at the end of each right session (i.e., each session between $\mathcal{A}$ and $\mathcal{O}$), the oracle $\mathcal{O}$ returns $\mathsf{Value}_\Gamma(\boldsymbol{s}^{\mathrm{CEC}})$ to $\mathcal{A}$ rather than $\mathsf{Value}_\Gamma(\boldsymbol{s})$ as the committed value of this session, where $\boldsymbol{s}=(s_1,\ldots,s_{10n})$ is the shares that are committed to in the row of Com in Stage 2, $\boldsymbol{s}^{\mathrm{CEC}}=(s_1^{\mathrm{CEC}},\ldots,s_{10n}^{\mathrm{CEC}})$ is the shares that are committed to in the row of CECom in Stage 3, and Γ is the subset that is committed to in the $\mathsf{CCACom}^{1:1}$ commitment in Stage 1.

Hybrid $D_2(n,y,z)$: Hybrid $D_2(n,y,z)$ is the same as $D_1(n,y,z)$ except for syntactical differences: Roughly speaking, $D_2(n,y,z)$ is an experiment in which $D_1(n,y,z)$ is executed in such a way that we can use the robust concurrent extraction lemma (Lemma 2.1) later. Formally, $D_2(n,y,z)$ is defined as follows. Recall that in the setting of the robust concurrent extraction lemma, an adversary, $\mathcal{A}_{\mathrm{robust}}$, launches the robust-concurrent attack by interacting with the online extractor $\mathcal{E}$; specifically, $\mathcal{A}_{\mathrm{robust}}$ interacts with $\mathcal{E}$ as a party A of an arbitrary two-party protocol Π while interacting with $\mathcal{E}$ as the committers of CECom concurrently and obtaining a value from $\mathcal{E}$ at the end of each session of CECom (where the values that are returned from $\mathcal{E}$ are supposed to be the committed values of the CECom sessions). Then, consider the following Π and $\mathcal{A}_{\mathrm{robust}}$.

- **Π:** In Π, the κ-round PPT ITM B (for which we are proving robustness of CCACom) interacts with party A honestly.
- **$\mathcal{A}_{\mathrm{robust}}$:** $\mathcal{A}_{\mathrm{robust}}$ takes a non-uniform advice z and internally executes $D_1(n,y,z)$ with the following changes. (Recall that the execution of $D_1(n,y,z)$ involves an interaction with $\mathcal{A}$.)

- In the session between $\mathcal{A}$ and B, $\mathcal{A}_{\text{robust}}$ forwards all the messages from $\mathcal{A}$ to $\mathcal{E}$ (who internally emulates B) and forwards back all the messages from $\mathcal{E}$ to $\mathcal{A}$.
- In Stage 3 of each right session, $\mathcal{A}_{\text{robust}}$ receives a row of CECom commitments from $\mathcal{A}$ and forwards it to $\mathcal{E}$ (who internally emulates the receivers of CECom). Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ denote the responses from $\mathcal{E}$ at the end of the row of the CECom commitments.
- At the end of each right session, $\mathcal{A}_{\text{robust}}$ sends $\mathsf{Value}_\Gamma(\boldsymbol{\alpha})$ to $\mathcal{A}$ as the committed value of this right session.

The output of $\mathcal{A}_{\text{robust}}$ is that of the internally emulated $\mathcal{A}$.

From the robust concurrent extraction lemma, there exists a robust simulator $\mathcal{S}_{\text{robust}}$ such that for the above $\mathcal{A}_{\text{robust}}$, there exists an online extractor $\mathcal{E}$ that satisfies the following.

- For any row of CECom that $\mathcal{A}_{\text{robust}}$ sends to $\mathcal{E}$, let $\boldsymbol{s}^{\text{CEC}} = (s_1^{\text{CEC}}, \ldots, s_{10n}^{\text{CEC}})$ be the shares that are committed to in this row of CECom and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ be the responses from $\mathcal{E}$ at the end of this row. Then, for every $j \in [10n]$, if the j-th CECom commitment in this row is valid and its committed value is uniquely determined, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10n})$ satisfies $\alpha_j = s_j^{\text{CEC}}$.
- $\mathcal{S}_{\text{robust}}$ can simulate the robust-concurrent attack between $\mathcal{A}_{\text{robust}}$ and $\mathcal{E}$.

Hybrid $D_2(n, y, z)$ is the experiment $\text{REAL}_{\mathcal{E},\Pi}^{\mathcal{A}_{\text{robust}}}(n, y, z)$ of the robust concurrent extraction lemma. The output of $D_2(n, y, z)$ is that of the internally emulated $\text{REAL}_{\mathcal{E},\Pi}^{\mathcal{A}_{\text{robust}}}(n, y, z)$.

Hybrid $D_3(n, y, z)$: Hybrid $D_3(n, y, z)$ differs from $D_2(n, y, z)$ in that the execution of $\text{REAL}_{\mathcal{E},\Pi}^{\mathcal{A}_{\text{robust}}}(n, y, z)$ (i.e., the robust-concurrent attack between $\mathcal{A}_{\text{robust}}$ and $\mathcal{E}$) is replaced with an interaction between party B of Π and the robust simulator $\mathcal{S}_{\text{robust}}$ of the robust concurrent extraction lemma. The output of $D_3(n, y, z)$ is the joint output of B and $\mathcal{S}_{\text{robust}}$.

For $k \in \{0, \ldots, 3\}$, let $\mathsf{D}_k(n, y, z)$ be the random variable for the output of $D_k(n, y, z)$.

Our simulator $\mathcal{S}$ is the simulator $\mathcal{S}_{\text{robust}}$ in $D_3(n, y, z)$. Notice that from the constructions of the hybrids, we have

$$\mathsf{D}_0(n, y, z) = \mathsf{output}_{B,\mathcal{A}^{\mathcal{O}}}\left[B(1^n, y) \leftrightarrow \mathcal{A}^{\mathcal{O}}(1^n, z)\right] \ ,$$
$$\mathsf{D}_3(n, y, z) = \mathsf{output}_{B,\mathcal{S}}\left[B(1^n, y) \leftrightarrow \mathcal{S}(1^n, z)\right] \ .$$

First, we notice that we have $\{\mathsf{D}_0(n, y, z)\}_{n\in\mathbb{N},y,z\in\{0,1\}^n} \overset{s}{\approx} \{\mathsf{D}_1(n, y, z)\}_{n\in\mathbb{N},y,z\in\{0,1\}^n}$. This is because from the same argument as in the proof of Claim 6.7, we can show that the view of $\mathcal{A}$ in $D_1(n, y, z)$ is statistically close to that in $D_0(n, y, z)$.

Next, we notice that we have $\{\mathsf{D}_1(n, y, z)\}_{n\in\mathbb{N},y,z\in\{0,1\}^n} \overset{s}{\approx} \{\mathsf{D}_2(n, y, z)\}_{n\in\mathbb{N},y,z\in\{0,1\}^n}$. This is because from the same argument as in the proof of Claim 6.8, we can show that an execution of $D_1(n, y, z)$ is statistically simulated in $D_2(n, y, z)$.

Next, we notice that we have $\{\mathsf{D}_2(n,y,z)\}_{n\in\mathbb{N},y,z\in\{0,1\}^n} \overset{s}{\approx} \{\mathsf{D}_3(n,y,z)\}_{n\in\mathbb{N},y,z\in\{0,1\}^n}$. This is because from the robust concurrent extraction lemma, we can show that the interaction between $\mathcal{S}_{\text{robust}}$ and B in $D_3(n,y,z)$ is statistically close to that between $\mathcal{A}_{\text{robust}}$ and $\mathcal{E}$ in $D_2(n,y,z)$.

Now, from the hybrid argument, we obtain Indistinguishability (6.15). □

Combining Lemmas 6.4 and 6.5, we obtain Theorem 6.1. This concludes the proof of Theorem 6.1. □

6.6 Black-Box Composable MPC Protocol

In this section, we show our black-box construction of a general MPC protocol. Our protocol is secure in the angel-based UC framework [PS04, CLP10, CLP16]. Roughly speaking, this framework (called the $\mathcal{H}$-EUC framework) is the same as the UC framework [Can01] except that both the adversary and the environment in the real and ideal worlds have access to a super-polynomial-time functionality $\mathcal{H}$ called an *angel* (or a *helper*). For details, see [PS04, CLP10, CLP16].

We use the results of Canetti et al. [CLP10, CLP16] and Lin and Pass [LP12]. Let $\langle C,R\rangle$ be any R_{CCA}-round robust CCA-secure commitment scheme, $\langle S,R\rangle$ be any R_{OT}-round semi-honest oblivious transfer protocol, and $\mathcal{H}$ be a helper that breaks $\langle C,R\rangle$ in essentially the same way as the committed-value oracle of $\langle C,R\rangle$ does. Then, Lin and Pass [LP12] showed that there exists a black-box $O(\max(R_{\mathsf{OT}},R_{\mathsf{CCA}}))$-round protocol that securely realizes the ideal oblivious transfer functionality $\mathcal{F}_{OT}$ in the $\mathcal{H}$-EUC framework.

Theorem 6.2 ([LP12])**.** *Assume the existence of an R_{CCA}-round robust CCA-secure commitment scheme $\langle C,R\rangle$ and the existence of an R_{OT}-round semi-honest oblivious transfer protocol $\langle S,R\rangle$. Then, there exists an $O(\max(R_{\mathsf{CCA}},R_{\mathsf{OT}}))$-round protocol that $\mathcal{H}$-EUC-realizes $\mathcal{F}_{OT}$. Furthermore, this protocol uses $\langle C,R\rangle$ and $\langle S,R\rangle$ only in a black-box way.*

In [CLP10, CLP16], Canetti et al. showed the following.

Theorem 6.3 ([CLP10, CLP16])**.** *For every well-formed functionality $\mathcal{F}$, there exists a constant-round $\mathcal{F}_{OT}$-hybrid protocol that $\mathcal{H}$-EUC-realizes $\mathcal{F}$.*

Then, we obtain the following theorem by combining Theorems 6.1, 6.2, and 6.3.

Theorem 6.4 (restatement of Main Theorem)**.** *Assume the existence of R_{OT}-round semi-honest oblivious transfer protocols. Then, there exists a super-polynomial-time helper $\mathcal{H}$ such that for every well-formed functionality $\mathcal{F}$, there exists a $\max(\widetilde{O}(\log^2 n),O(R_{\mathsf{OT}}))$-round protocol that $\mathcal{H}$-EUC-realizes $\mathcal{F}$. Furthermore, this protocol uses the underlying oblivious transfer protocol only in a black-box way.*

Commit Phase

The committer C and the receiver R receive common inputs 1^n and $\mathsf{id} \in \{0,1\}^{2^{t(n)-1}}$. To commit to $v \in \{0,1\}$, the committer C chooses random $v_1, \ldots, v_{2^{t(n)-1}} \in \{0,1\}^n$ such that $v = \bigoplus_j v_j$, and for each $j \in [2^{t(n)-1}]$ in parallel, C^* commits to v_j by using CCACom with tag (j, id_j), where id_j is the j-th bit of id.

Decommit Phase

C sends v to R and decommits all the CCACom commitments.

Figure 6.13: One-one CCA-secure commitment scheme $\mathsf{CCACom}^{1:1}$.

6.7 Appendix to Chapter 6

6.7.1 One-One CCA Commitment for Long Tags from Parallel CCA Commitment for Short Tags

Lemma 6.6. *Let $r(\cdot)$ and $t(\cdot)$ be arbitrary functions such that $t(n) = O(\log n)$, and let CCACom be an $r(n)$-round commitment scheme that satisfies strong computational binding property and parallel CCA security for tags of length $t(n)$. Then, there exists an $r(n)$-round commitment scheme $\mathsf{CCACom}^{1:1}$ that satisfies strong computational binding property and one-one CCA security for tags of length $2^{t(n)-1}$. Furthermore, if CCACom uses the underlying one-way function only in a black-box way, so does $\mathsf{CCACom}^{1:1}$.*

Proof. $\mathsf{CCACom}^{1:1}$ is shown in Figure 6.13. The strong computational binding property follows from that of CCACom. Thus, it remains to show that $\mathsf{CCACom}^{1:1}$ is one-one CCA secure.

We show that for any PPT adversary $\mathcal{A}$ that interacts with O only in a single session, the following are computationally indistinguishable:

- $\{\mathsf{IND}_0(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*}$
- $\{\mathsf{IND}_1(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)\}_{n\in\mathbb{N}, z\in\{0,1\}^*}$

Without loss of generality, we can assume that the tag that $\mathcal{A}$ uses in the right session is always different from the tag that $\mathcal{A}$ uses in the left session. (This is because instead of using the same tag in the left and right sessions, $\mathcal{A}$ can use different tags in the left and right sessions and then output $\bot$; recall that the output of the experiment is $\bot$ whenever $\mathcal{A}$ uses the same tag in the left and right sessions.)

Assume for contradiction that there exist a PPT distinguisher $\mathcal{D}$ and a polynomial $p(\cdot)$ such that for infinitely many n, there exists $z \in \{0,1\}^*$ such that $\mathcal{D}$ distinguishes $\mathsf{IND}_0(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)$ and $\mathsf{IND}_1(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)$ with advantage at least $1/p(n)$. In the following, we fix any such n and z.

Let us consider the following PPT adversary $\mathcal{B}$ against CCA security of CCACom. $\mathcal{B}$ internally invokes $\mathcal{A}$ and simulates the experiment $\mathsf{IND}_0(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)$ for $\mathcal{A}$ as follows. First, $\mathcal{B}$ chooses random $j^* \in [2^{t(n)-1}]$, and for each $j \in [2^{t(n)-1}] \setminus \{j^*\}$,

$\mathcal{B}$ chooses random $v_j \in \{0,1\}^n$. Then, in the left session, when $\mathcal{A}$ outputs challenge values $m_0, m_1 \in \{0,1\}^n$ and tag $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_{2^{t(n)-1}})$, $\mathcal{B}$ sets $v_{j^*}^{(b)} := m_b \oplus \bigoplus_{j \neq j^*} v_j$ for each $b \in \{0,1\}$ and sends challenge $v_{j^*}^{(0)}, v_{j^*}^{(1)}$ and tag $(j^*, \mathsf{id}_{j^*}) \in \{0,1\}^{t(n)}$ to the external left committer. When $\mathcal{B}$ receives a CCACom commitment from the left committer (the committed value is either $v_{j^*}^{(0)}$ or $v_{j^*}^{(1)}$), $\mathcal{B}$ forwards it to $\mathcal{A}$. At the same time, $\mathcal{B}$ generates CCACom commitments to $(v_j)_{j \neq j^*}$ and sends them to $\mathcal{A}$. In the right session, $\mathcal{B}$ forwards a $\mathsf{CCACom}^{1:1}$ commitment from $\mathcal{A}$ to $\mathcal{O}$ as $2^{t(n)-1}$ parallel commitments of CCACom with tags $\{(j, \widetilde{\mathsf{id}}_j)\}_{j \in [2^{t(n)-1}]}$. Then, $\mathcal{B}$ receives $(v_1, \ldots, v_{2^{t(n)-1}})$ from $\mathcal{O}$, and if $v_j \neq \bot$ for all $j \in [2^{t(n)-1}]$, $\mathcal{B}$ returns $v := \bigoplus_j v_j$ to $\mathcal{A}$; if $v_j = \bot$ for some j, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}$. Let y be the output of the simulated experiment $\mathsf{IND}_0(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)$, and let $\beta \leftarrow \mathcal{D}(y)$. Then, if $\mathsf{id}_{j^*} = \widetilde{\mathsf{id}}_{j^*}$, $\mathcal{B}$ outputs fail, and otherwise, $\mathcal{A}$ outputs fail with probability $(N-1)/N$ and outputs β with probability $1/N$, where $N = |\{j \text{ s.t. } \mathsf{id}_j \neq \widetilde{\mathsf{id}}_j\}|$ is the Hamming distance between id and $\widetilde{\mathsf{id}}$.

We reach a contradiction by showing that $\mathcal{B}$ breaks the CCA security of CCACom; in particular,

$$\left|\Pr\left[\mathsf{IND}_0(\mathsf{CCACom}, \mathcal{B}, n, z) = 1\right] - \Pr\left[\mathsf{IND}_1(\mathsf{CCACom}, \mathcal{B}, n, z) = 1\right]\right| \geq \frac{1}{p(n) \cdot \mathsf{poly}(n)} \ .$$

For $b \in \{0,1\}$, let β_b be the random variable representing the value of β in $\mathsf{IND}_b(\mathsf{CCACom}, \mathcal{B}, n, z)$ and abort_b be the event that $\mathcal{B}$ outputs fail in $\mathsf{IND}_b(\mathsf{CCACom}, \mathcal{B}, n, z)$. Since $\mathcal{B}$ internally simulates $\mathsf{IND}_0(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)$ or $\mathsf{IND}_1(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)$ perfectly depending on the value that is committed to in the left session, we have

$$\Pr\left[\beta_b = 1\right] = \Pr\left[\mathcal{D}(\mathsf{IND}_b(\mathsf{CCACom}^{1:1}, \mathcal{A}, n, z)) = 1\right] \ .$$

Hence, from our assumption, we have

$$\left|\Pr\left[\beta_0 = 1\right] - \Pr\left[\beta_1 = 1\right]\right| \geq \frac{1}{p(n)} \ .$$

Also, since we assume that it always holds that $\mathsf{id} \neq \widetilde{\mathsf{id}}$, for each $b \in \{0,1\}$ we have

$$\Pr\left[\neg\mathsf{abort}_b\right] = \frac{N}{2^{t(n)-1}} \cdot \frac{1}{N} = \frac{1}{2^{t(n)-1}} \ .$$

Note that when $\mathsf{id}_{j^*} \neq \widetilde{\mathsf{id}}_{j^*}$, a tag $(j, \widetilde{\mathsf{id}}_j)$ in the right session is different from the tag (j^*, id_{j^*}) in the left session for each $j \in [2^{t(n)-1}]$. Hence, when abort_b does not occur, the output of $\mathsf{IND}_b(\mathsf{CCACom}, \mathcal{B}, n, z)$ is β_b. Thus, we have

$$\begin{aligned}
&\left|\Pr\left[\mathsf{IND}_0(\mathsf{CCACom}, \mathcal{B}, n, z) = 1\right] - \Pr\left[\mathsf{IND}_1(\mathsf{CCACom}, \mathcal{B}, n, z) = 1\right]\right| \\
&= \left|\Pr\left[\beta_0 = 1 \wedge \neg\mathsf{abort}_0\right] - \Pr\left[\beta_1 = 1 \wedge \neg\mathsf{abort}_1\right]\right| \\
&= \left|\Pr\left[\beta_0 = 1\right] - \Pr\left[\beta_1 = 1\right]\right| \cdot \frac{1}{2^{t(n)-1}} \\
&\geq \frac{1}{p(n) \cdot \mathsf{poly}(n)} \ .
\end{aligned}$$

In the third line, we use $\Pr[\beta_b = 1 \wedge \neg \mathsf{abort}_b] = \Pr[\beta_b = 1] \cdot \Pr[\neg \mathsf{abort}_b]$ (i.e., the independence between the event abort_b and the event that $\beta_b = 1$, which follows from the fact that abort_b always occurs with probability $1/2^{t(n)-1}$, independently of the values of id and $\widetilde{\mathsf{id}}$). This concludes the proof. □

Chapter 7

Conclusion

In this thesis, we studied zero-knowledge proofs and secure computation, two of the most fundamental protocols in cryptography. Zero-knowledge proofs are counter-intuitive protocols that allow provers to convince verifiers of the correctness of statements without revealing any additional knowledge about the statements, and secure computation protocols are powerful protocols that enable mutually distrustful parties to jointly compute any functionalities on their secret inputs without compromising the correctness of the outputs and the privacy of the inputs. Zero-knowledge proofs are fundamental in cryptography because they are used as key building blocks in numerous other protocols, and secure computation protocols are fundamental in cryptography because their powerful generality allows us to obtain strong feasibility results about cryptographic protocols.

Our focus was to obtain new constructions that satisfy two strong security notions, concurrent security and leakage resilience. As discussed in Chapter 1, the main advantage of achieving strong security is that it enables us to use protocols in more broad applications—in the case of concurrent security and leakage-resilience, the former allows us to use protocols in a setting where multiple instances of the protocols are executed concurrently, and the latter allows us to use protocols in the setting where adversaries might employ side-channel attacks on physical implementations of the protocols.

The result of this thesis was four results about zero-knowledge proofs and secure computation with strong security guarantees. A common motivation behind these results is to achieve concurrent security and leakage resilience at as low additional cost as possible, where the cost is defined in terms of efficiency and hardness assumptions; in other words, our results concern the problem of constructing zero-knowledge protocols and secure computation that satisfy concurrent security or leakage resilience with good efficiency under weak hardness assumptions.

Concretely, we showed the following four results in this thesis.

1. In Chapter 3, we constructed a statistical concurrent non-malleable zero-knowledge argument from one-way functions, where statistical concurrent non-malleable zero-knowledge is currently the strongest notion of concurrent security for zero-knowledge protocols in the plain model (i.e., in the model without any help from trusted third parties). Since the existence of one-way functions is the weakest hardness assumption from which (standard) zero-knowledge ar-

guments are known to be obtained, this result implies that statistical concurrent non-malleable zero-knowledge arguments can be obtained at no additional cost in terms of hardness assumptions.

2. In Chapter 4, we constructed a constant-round leakage-resilient zero-knowledge argument from collision-resistant hash functions. Since the existing constructions either have more than constant number of rounds [GJS11] or rely on an assumption that is seemingly much stronger than the existence of collision-resistant hash functions [Pan14], this result implies that leakage-resilient zero-knowledge arguments with asymptotically optimal round complexity can be obtained under much weaker assumptions than previously known. Constructing them from even weaker assumptions, such as the existence of one-way functions, is an interesting open question.

3. In Chapter 5, we constructed a concurrent zero-knowledge argument that has a new non-black-box simulation proof of security. Since non-black-box simulation is proven to be essential for solving the long-standing open question of constructing constant-round concurrent zero-knowledge arguments [CKPR02], and our non-black-box simulation technique for concurrent zero-knowledge is arguably simpler than the existing one [Goy13], this result can be a useful starting point for future research. Using our technique to obtain a constant-round (or even sub-polynomial-round) concurrent zero-knowledge argument is the biggest open problem that is left by this thesis.

4. In Chapter 6, we constructed a polylogarithmic-round concurrently secure multiparty computation protocol that makes only black-box use of the underlying cryptographic primitives. Since the existing constructions that satisfy the same security notion as ours (namely, angel-based UC security) either make non-black-box use of underlying primitives (and thus are typically inefficient) [PS04, MMY06, CLP10, CLP16, GLP+15] or have polynomial number of rounds [LP12], this result implies that concurrently secure computation can be obtained in a black-box way with much smaller round complexity than previously known.

We notice that, at a high level, our results can be viewed as a step to solve a fundamental problem about concurrent security and leakage resilience, that is, the problem of constructing secure computation protocols that satisfy concurrent security and leakage resilience with optimal efficiency under minimum assumptions. (This problem is fundamental because by solving it, we can obtain a strong feasibility result about concurrent security and leakage resilience thorough the generality of secure computation.) Indeed, it is easy to see that each of our results concerns a natural simplified version of this fundamental problem, where the simplification is

- to focus on zero-knowledge protocols rather than secure computation,[48] and/or

[48] Studying a security notion on zero-knowledge protocols is a natural first step to study it on secure computation since zero-knowledge protocols are key building blocks of existing secure computation protocols.

- to focus on either concurrent security or leakage resilience (i.e., not to consider them simultaneously).

When viewed as a step to solve this fundamental problem, our results made significant advances on its natural simplified versions by reducing required cost in terms of efficiency and hardness assumptions:

- **Case 1. Focus on zero-knowledge protocols and concurrent security:** Our first result implies that on zero-knowledge protocols, even the strongest notion of concurrent security can be achieved under a minimum assumption.

- **Case 2. Focus on zero-knowledge protocols and leakage resilience:** Our second result implies that on zero-knowledge protocols, leakage resilience can be achieved with asymptotically optimal round complexity under a much weaker hardness assumption than previously known.

- **Case 3. Focus on secure computation and concurrent security:** Our fourth result implies that on secure computation, a popular notion of concurrent security (namely angel-based UC security) can be achieved without using inefficient non-black-box uses of underlying cryptographic primitives and large round complexity.

Additionally, as we discuss below, our third result has a potential to become a useful starting point for removing a main obstacle about this fundamental problem.

Future Directions

A natural future direction is to pursue feasibility results about concurrent security and leakage resilience further. In other words, a natural direction is to try to strengthen the results of this thesis in order to solve the aforementioned fundamental problem of constructing secure computation protocols that satisfy concurrent security and leakage resilience with optimal efficiency under minimum assumptions.

Regarding this direction, there are (at least) three major problems that are not addressed in this thesis.

- **Achieving concurrent security with optimal round complexity:** The first one is the problem of improving the round complexity of concurrent zero-knowledge protocols. As mentioned in Chapter 5, the state-of-the-art of this problem is $\omega(\log n)$ rounds [PRS02], and the construction of constant-round concurrent zero-knowledge protocols under standard hardness assumptions is considered to be a big open question about concurrent security (indeed, this open question is currently a major obstacle to achieve concurrent security in constant number of rounds on many other cryptographic protocols; for example, concurrently secure computation currently requires logarithmic complexity because concurrent zero-knowledge requires logarithmic round complexity).

  An approach towards this problem is to improve our new non-black-box simulation technique in the third result of this thesis so that it requires fewer rounds

than what it currently requires. (Recall that, as mentioned in Chapter 5, non-black-box simulation techniques are necessary to improve the state-of-the-art of the round complexity of concurrent zero-knowledge protocols.) Another, more ambitious, approach is to develop a new non-black-box simulation technique that is completely different from the existing ones [Bar01, BP15].

- **Achieving leakage resilience on secure computation:** The second one is the problem of constructing leakage-resilient (or leakage-tolerant[49]) secure computation protocols. There already exist several works that study leakage-resilient secure computation protocols [BGJK12, BGJ+13, BDL14], but their results have a drawback that either leakage-free preprocessing phases are required or only a weaker security notion is achieved (see [BDL14] for details). To the best of our knowledge, construing secure computation protocols that do not have this drawback is an open question.

  A potential approach towards this problem is to use the techniques that we developed for leakage-resilient zero-knowledge protocols in the second result of this thesis. Another potential approach is to prove an impossibility result about leakage-resilient secure computation (it is already shown that, as long as the security is proven thorough black-box simulation techniques, interactive leakage-free preprocessing is necessary for leakage-resilient secure computation [OPV15]).

- **Achieving concurrent security and leakage resilience simultaneously:** The third one is the problem of achieving concurrent security and leakage resilience simultaneously. It is known that one can achieve concurrent security and leakage resilience simultaneously on several cryptographic protocols and even on secure computation in the UC framework [BCH12, BGJ+13, BDL14]; however, these results are not quite satisfactory because UC security has a drawback that it cannot be achieved in the plain model (cf. Section 6.1) and indeed these results are shown in a model where a little help from trusted third party is available.[50] To the best of our knowledge, the problem of achieving concurrent security and leakage resilience simultaneously on zero-knowledge protocols and secure computation in the plain model is an open question.

  A potential approach towards this problem is to focus on zero-knowledge proofs and try to combine existing concurrent zero-knowledge protocols (such as that in Chapter 3) with existing leakage-resilient zero-knowledge protocols (such as that in Chapter 4).

Another natural future direction is to study zero-knowledge proofs and secure computation that satisfy other strong security notions. An interesting candidate is resettable security (cf. Section 1.2), which has been studied for both zero-knowledge proofs and security computation [CGGM00, BGGL01, DGS09, GS09] and is deeply related to concurrent security (for example, it is known that resettable zero-knowledge implies concurrent zero-knowledge [CGGM00]).

[49] See Footnote 21 in Section 4.1.

[50] Additionally, the result about secure computation in [BDL14] is shown in a model where leakage-free preprocessing is available

Acknowledgment

First and foremost, I would like to thank Prof. Tatsuaki Okamoto for his invaluable guidance since the very beginning of my career as a cryptographer. He was a supervisor when I was a master-course student and now is a colleague at NTT, and I feel extremely fortunate to have had the opportunity to learn from his remarkably deep and broad knowledge and perspective about cryptography. I also would like to thank him for serving on my thesis committee; without his supports, I could not have written this thesis.

I also would like to thank Prof. Yoshifumi Manabe (now at Kogakuin University) for his guidance as a co-supervisor during my master-course studies. I am especially grateful to him for having a meeting with me almost every week and listening patiently to all of my questions during my master-course studies. Thinking that he worked at NTT at that time and thus must have been very busy (I can now guess how busy he must have been from my own experience of working at NTT), I feel deep gratitude for continuously finding time just for having meetings with me.

I also would like to express my deep gratitude to Prof. Toru Ishida, Prof. Yoshimasa Nakamura, and Prof. Yasuo Okabe for serving on my thesis committee. Their advice and comments greatly helped me write this thesis.

Special thanks go to the members of the NTT crypto team. Working with great and hardworking cryptographers like them is really delightful and always motivates me. I also thank them for allowing me to focus on my own research since my first year at NTT—all the results in this thesis were obtained after I joined NTT.

Finally, I would like to thank my family members. Without their continuous help, my life would have been completely different.

Bibliography

[AGP14] Prabhanjan Ananth, Vipul Goyal, and Omkant Pandey. Interactive proofs under continual memory leakage. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 164–182. Springer, Heidelberg, August 2014.

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance: A cautionary note. In *WOEC*, pages 1–11, 1996.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM*, 45(1):70–122, January 1998.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001.

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd FOCS*, pages 345–355. IEEE Computer Society Press, November 2002.

[Bar05] Boaz Barak. How to play almost any mental game over the net - Concurrent composition via super-polynomial simulation. In *46th FOCS*, pages 543–552. IEEE Computer Society Press, October 2005.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCH12] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 266–284. Springer, Heidelberg, March 2012.

[BCY91] Gilles Brassard, Claude Crépeau, and Moti Yung. Constant-round perfect zero-knowledge computationally convincing protocols. *Theoretical Computer Science*, 84(1):23–52, July 1991.

[BDL14] Nir Bitansky, Dana Dachman-Soled, and Huijia Lin. Leakage-tolerant computation with input-independent preprocessing. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 146–163. Springer, Heidelberg, August 2014.

[Bea92] Donald Beaver. Foundations of secure interactive computing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 377–391. Springer, Heidelberg, August 1992.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *42nd FOCS*, pages 116–125. IEEE Computer Society Press, October 2001.

[BGJ+13] Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 316–334. Springer, Heidelberg, August 2013.

[BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1235–1254. ACM Press, May 2012.

[BJY97] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 280–305. Springer, Heidelberg, May 1997.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *the International Congress of Mathematicians*, pages 1444–1451, 1986.

[BP12] Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *53rd FOCS*, pages 223–232. IEEE Computer Society Press, October 2012.

[BP13] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 241–250. ACM Press, June 2013.

[BP15] Nir Bitansky and Omer Paneth. On non-black-box simulation and the impossibility of approximate obfuscation. *SIAM Journal on Computing*, 44(5):1325–1383, 2015.

[BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *47th FOCS*, pages 345–354. IEEE Computer Society Press, October 2006.

[BY03] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 1–18. Springer, Heidelberg, April 2003.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CDMW08] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 427–444. Springer, Heidelberg, March 2008.

[CDMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 387–402. Springer, Heidelberg, March 2009.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *32nd ACM STOC*, pages 235–244. ACM Press, May 2000.

[CKL06] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *Journal of Cryptology*, 19(2):135–167, April 2006.

[CKPR02] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. *SIAM Journal on Computing*, 32(1):1–47, 2002.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st FOCS*, pages 541–550. IEEE Computer Society Press, October 2010.

[CLP13a] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 80–99. Springer, Heidelberg, March 2013.

[CLP13b] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from P-certificates. In *54th FOCS*, pages 50–59. IEEE Computer Society Press, October 2013.

[CLP15] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 287–307. Springer, Heidelberg, August 2015.

[CLP16] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. *SIAM Journal on Computing*, 45(5):1793–1834, 2016.

[COSV17] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 711–742. Springer, Heidelberg, November 2017.

[CPS13] Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 231–240. ACM Press, June 2013.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DGS09] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *50th FOCS*, pages 251–260. IEEE Computer Society Press, October 2009.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *Journal of the ACM*, 51(6):851–898, 2004.

[DPP98] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, 1998.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[FS90a] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990.

[FS90b] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 526–544. Springer, Heidelberg, August 1990.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 99–116. Springer, Heidelberg, April 2012.

[GGS15] Vipul Goyal, Divya Gupta, and Amit Sahai. Concurrent secure computation via non-black box simulation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 23–42. Springer, Heidelberg, August 2015.

[GJO+13] Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, and Ivan Visconti. Constant-round concurrent zero knowledge in the bounded player model. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2013.

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 297–315. Springer, Heidelberg, August 2011.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GL91] Shafi Goldwasser and Leonid A. Levin. Fair computation of general functions in presence of immoral majority. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 77–93. Springer, Heidelberg, August 1991.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012.

[GLP+15] Vipul Goyal, Huijia Lin, Omkant Pandey, Rafael Pass, and Amit Sahai. Round-efficient concurrently composable secure computation via a robust extraction lemma. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 260–289. Springer, Heidelberg, March 2015.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMOS07] Vipul Goyal, Ryan Moriarty, Rafail Ostrovsky, and Amit Sahai. Concurrent statistical zero-knowledge arguments for NP from one way functions. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2007.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 695–704. ACM Press, June 2011.

[Goy13] Vipul Goyal. Non-black-box simulation in the fully concurrent setting. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 221–230. ACM Press, June 2013.

[GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 54–71. Springer, Heidelberg, April 2009.

[GST17] Daniel Genkin, Adi Shamir, and Eran Tromer. Acoustic cryptanalysis. *Journal of Cryptology*, 30(2):392–443, April 2017.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 412–426. Springer, Heidelberg, March 2008.

[HIK+11] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM Journal on Computing*, 40(2):225–266, 2011.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 408–423. Springer, Heidelberg, August 1998.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In Jon M. Kleinberg, editor, *38th ACM STOC*, pages 99–108. ACM Press, May 2006.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th FOCS*, pages 230–235. IEEE Computer Society Press, October / November 1989.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

[Kiy14] Susumu Kiyoshima. Round-efficient black-box construction of composable multi-party computation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 351–368. Springer, Heidelberg, August 2014.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC press, 2014.

[KMO14] Susumu Kiyoshima, Yoshifumi Manabe, and Tatsuaki Okamoto. Constant-round black-box construction of composable multi-party computation protocol. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 343–367. Springer, Heidelberg, February 2014.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In *33rd ACM STOC*, pages 560–569. ACM Press, July 2001.

[LP09] Huijia Lin and Rafael Pass. Non-malleability amplification. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 189–198. ACM Press, May / June 2009.

[LP11a] Huijia Lin and Rafael Pass. Concurrent non-malleable zero knowledge with adaptive inputs. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 274–292. Springer, Heidelberg, March 2011.

[LP11b] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 705–714. ACM Press, June 2011.

[LP12] Huijia Lin and Rafael Pass. Black-box constructions of composable protocols without set-up. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 461–478. Springer, Heidelberg, August 2012.

[LP15] Huijia Lin and Rafael Pass. Constant-round nonmalleable commitments from any one-way function. *Journal of the ACM*, 62(1):5:1–5:30, March 2015.

[LPTV10] Huijia Lin, Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable zero knowledge proofs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 429–446. Springer, Heidelberg, August 2010.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 571–588. Springer, Heidelberg, March 2008.

[LZ11] Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *Journal of Cryptology*, 24(4):761–799, October 2011.

[MMY06] Tal Malkin, Ryan Moriarty, and Nikolai Yakovenko. Generalized environmental security from number theoretic assumptions. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 343–359. Springer, Heidelberg, March 2006.

[MOSV06] Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil P. Vadhan. Concurrent zero knowledge without complexity assumptions. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 1–20. Springer, Heidelberg, March 2006.

[MR92] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 392–404. Springer, Heidelberg, August 1992.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, 1998.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.

[OOR+14] Claudio Orlandi, Rafail Ostrovsky, Vanishree Rao, Amit Sahai, and Ivan Visconti. Statistical concurrent non-malleable zero knowledge. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 167–191. Springer, Heidelberg, February 2014.

[OPV10] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency preserving transformations for concurrent non-malleable zero knowledge. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 535–552. Springer, Heidelberg, February 2010.

[OPV15] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Impossibility of black-box simulation against leakage attacks. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 130–149. Springer, Heidelberg, August 2015.

[OW93] Rafail Ostrovsky Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *2nd ISTCS*, pages 3–17. IEEE Computer Society, June 1993.

[Pan14] Omkant Pandey. Achieving constant round leakage-resilient zero-knowledge. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 146–166. Springer, Heidelberg, February 2014.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003.

[PLV12] Rafael Pass, Huijia Lin, and Muthuramakrishnan Venkitasubramaniam. A unified framework for UC from only OT. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 699–717. Springer, Heidelberg, December 2012.

[PPS15] Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 638–667. Springer, Heidelberg, March 2015.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *46th FOCS*, pages 563–572. IEEE Computer Society Press, October 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 533–542. ACM Press, May 2005.

[PR08] Rafael Pass and Alon Rosen. New and improved constructions of nonmalleable cryptographic protocols. *SIAM Journal on Computing*, 38(2):702–752, 2008.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *43rd FOCS*, pages 366–375. IEEE Computer Society Press, November 2002.

[PRT13] Rafael Pass, Alon Rosen, and Wei-Lung Dustin Tseng. Public-coin parallel zero-knowledge for NP. *Journal of Cryptology*, 26(1):1–10, January 2013.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, *36th ACM STOC*, pages 242–251. ACM Press, June 2004.

[PTV14] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Concurrent zero knowledge, revisited. *Journal of Cryptology*, 27(1):45–66, January 2014.

[PTW09] Rafael Pass, Wei-Lung Dustin Tseng, and Douglas Wikström. On the composition of public-coin zero-knowledge protocols. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 160–176. Springer, Heidelberg, August 2009.

[PV08] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. On constant-round concurrent zero-knowledge. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 553–570. Springer, Heidelberg, March 2008.

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Heidelberg, March 2009.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 415–431. Springer, Heidelberg, May 1999.

[Ven14] Muthuramakrishnan Venkitasubramaniam. On adaptively secure protocols. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 455–475. Springer, Heidelberg, September 2014.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

List of Earlier Publications

This thesis is based on the following earlier publications.

Chapter 3.

Susumu Kiyoshima. Statistical concurrent non-malleable zero-knowledge from one-way functions. In Rosario Gennaro and Matthew J. B. Robshaw, editors, CRYPTO 2015, Part II, volume 9216 of LNCS, pages 85–106. Springer, Heidelberg, August 2015. ©IACR 2015, http://dx.doi.org/10.1007/978-3-662-48000-7_5.

Chapter 4.

Susumu Kiyoshima. Constant-round leakage-resilient zero-knowledge from collision resistance. In Marc Fischlin and Jean-Sébastien Coron, editors, EUROCRYPT 2016, Part II, volume 9666 of LNCS, pages 93–123. Springer, Heidelberg, May 2016. ©IACR 2016, http://dx.doi.org/10.1007/978-3-662-49896-5_4.

Chapter 5.

Susumu Kiyoshima. An alternative approach to non-black-box simulation in fully concurrent setting. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, TCC 2015, Part I, volume 9014 of LNCS, pages 290–318. Springer, Heidelberg, March 2015. ©IACR 2015, http://dx.doi.org/10.1007/978-3-662-46494-6_13.

Chapter 6.

Susumu Kiyoshima. Round-efficient black-box construction of composable multi-party computation. In Juan A. Garay and Rosario Gennaro, editors, CRYPTO 2014, Part II, volume 8617 of LNCS, pages 351–368. Springer, Heidelberg, August 2014. ©IACR 2014, http://dx.doi.org/10.1007/978-3-662-44381-1_20.