On Universally Composable KEM, DEM and Three Cryptographic Channels

Waka NAGAO

Abstract

The goal of this research is to define the non-malleability of a key encryption mechanism (KEM) and prove equivalence between the indistinguishability and non-malleability for KEM and the data encryption mechanism (DEM), respectively, which are ISO standards. In addition, this thesis proves the equivalence (or reducible properties) among three cryptographic channels, a secure channel (SC), an anonymous channel (AC), and a direction-indeterminable channel (DIC).

These days, we are surrounded by network infrastructures that provide us with information environments. In these network infrastructures, we communicate with friends and associates using convenient network systems such as the Internet, e-mail, and web phones. The issue of security has become increasingly important as the amount of network communications increases, i.e., private and individual information are exposed to many risks. Therefore, adequate security measures must be established in the current information society. The contribution of this thesis is to aid in providing security against many threats and risks facing the current information society. This thesis comprises three brief compositions: semantic security for KEM and DEM; universal composability for KEM and DEM; and universal composability for three cryptographic channels with Probabilistic Input Output Automata (PIOA).

1. **Semantic Security for KEM and DEM**

   First, we study the basic cryptographic security notion, semantic security, for KEM and DEM. KEM and DEM were introduced by Shoup to formalize asymmetric encryption specified for key distribution and symmetric encryption specified for data exchange in ISO standards. The system consists of KEM and DEM and enables a key delivery mechanism and message sending mechanism with a high level of security. Shoup defined “semantic security (IND) against adaptive chosen ciphertext attacks (CCA2)” as a desirable security notion of KEM, i.e., IND-CCA2 KEM. This thesis defines ”non-malleability (NM)” for KEM, which is a stronger security notion than IND. We provide three definitions of NM for KEM, and show that these three definitions are equivalent. We then prove that NM-CCA2 KEM is equivalent to IND-CCA2 KEM and that non-malleability against adaptive chosen plaintext/ciphertext attack (NM-P2-C2) DEM is equivalent to IND-P2-C2 DEM, respectively. More specifically, we show that NM is equivalent to IND for KEM under the CCA2 attack and that NM is equivalent to IND for DEM under the P2-C2 attack, although NM is stronger than IND.

2. **Universally Composable KEM and DEM**

   Second, we studied the universally composable (UC) framework for KEM and DEM. One of the essential frameworks for security composability, the UC framework, was introduced by Canetti. The framework investigates the composability of security functions such as public key encryption, authentication, and a secure channel. We define universally composable functions for KEM and DEM, and show that IND-CCA2 KEM (or NM-CCA2 KEM) is equivalent to UC KEM and that “IND against adaptive chosen plaintext / ciphertext attack (IND-P2-C2)” DEM is equivalent to UC DEM.

3. **Universally Composable Three Cryptographic Channels with PIOA**

   Third, we studied the relationship among three cryptographic channels, a SC, AC, and DIC using UC with the precise framework PIOA. The relationship among the three cryptographic channels was investigated by Okamoto. He showed that the three cryptographic channels are reducible to each other, but did not thoroughly consider communication schedules and composable security. This thesis refines the relationship among the three channels in light of the communication schedules and composable security. We model parties using the task-PIOA to treat communication schedules, and adopt the UC framework by Canetti to treat composable security. We show that a class of anonymous channels, two-anonymous channels (2AC), and DIC are reducible to each other under some types of schedules and that DIC and SC are reducible to each other under some types of schedules in the UC framework using the PIOA model.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Professor Tatsuaki Okamoto of Kyoto University and NTT Labs., for his continuous guidance, precise advice, and helpful discussions. His unwavering attitude toward research is exemplary and he is an outstanding role model. I also appreciate the continuous guidance and helpful comments I received from Associate Professor Yoshifumi Manabe of Kyoto University and NTT Labs. His kindness in taking the time to discuss my research and his helpful advice were invaluable.

I am very gratefully for the valuable comments I received from other members of my thesis committee at Kyoto University, Professor Hitoshi Nagamochi, Professor Tohru Ishida, and Associate Professor Mizuho Iwaihara.

I am grateful to my other adviser, Associate Professor Keisuke Tanaka at the Tokyo Institute of Technology, for his valuable advice and discussions.

Finally, I would like to thank the members of the Okamoto and Manabe Laboratory and all members of the Ishida and Matsubara Laboratory at Kyoto University.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Security is one of the most important notions in the current information-based society. These days, individuals take steps to protect personal information such names, addresses, and card numbers because this type of information is exposed to risk by strangers in the current network society. To protect this information and secret data, we must construct a security system or at least implement some rules. For example, a network system and application system that implements a high level of security should be constructed to protect against disingenuous attackers.

In the theoretical security paradigm, we employed symmetric key encryption until the 1960s and implemented public key encryption to protect information against network attackers in the 1970s. Symmetric key encryption has a problematic point when users must exchange the information regarding the key. However, public key encryption solves the key delivery difficulty.

The Key Encapsulation Mechanism (KEM) and Data Encapsulation Mechanism (DEM) were proposed by Shoup as ISO standards for hybrid-public-key encryption (H-PKE) [55]. The security notion of indistinguishability (IND) (or semantic security) for KEM and DEM was also defined by Shoup. On the other hand, a definition of another stronger security notion "non-malleability (NM)" was introduced by Katz and Yung for private-key encryption (or DEM) and they investigated the relations between IND and NM [37] (their results included proof that IND-P2-C2 is equivalent to NM-P2-C2 for private-key encryption).

In this thesis, we investigate two stronger security notions for KEM and DEM and research the equivalence (or reducible properties) among three cryptographic channels: secure channels, anonymous channels, and direction-indeterminable channels. More specifically, one of the two stronger security notions is a semantic secure NM for KEM and the other is UC for KEM and DEM.

NM for public-key encryption (PKE) was introduced [29, 3, 10] as a stronger security notion than IND and analogous definitions of NM for KEM were introduced in [41, 42]. Since the NM of PKE has been defined using a *message space* specified by an adversary, the existing NM definitions of KEM [41, 42] use a *key space* specified by an adversary, which corresponds to a message space for PKE. These existing NM definitions of KEM, however, are available only for a few types of KEM schemes, e.g., a KEM scheme constructed from a PKE where a random string plaintext to PKE is a session key output by KEM. This is because an adversary can specify a very small key space, e.g., $\{K_0, K_1\}$, but in a general type of KEM scheme, it may be hard for a polynomial-time machine (an experiment in NM definitions) to produce a ciphertext along with a key in this specified small key space as the output of the encryption function. In other words, the existing NM definitions cannot be used for such a general type of KEM scheme.

A weaker security notion of NM, wNM, was introduced and investigated by Herranz et al. [35]. The wNM-CCA2 KEM is unlikely to imply IND-CCA2 KEM. Therefore, wNM is not considered to be a feasible definition of the NM for KEM, since a feasible definition of NM(-ATK) should imply IND(-ATK) (ATK $\in$ {CPA, CCA1, CCA2}) (In fact, the standard definition of NM(-ATK) of PKE implies IND(-ATK)).

This thesis provides NM definitions that satisfy the following feasible requirements:

1. The NM definitions are available for any type of KEM scheme, in which no key space is used.

2. The NM definitions are stronger than IND, i.e., NM(-ATK) implies IND(-ATK)). For more detailed description on this matter, see Section 5.1.1 and Theorem 4).

3. The NM definitions capture the naive NM property that the adversary is given challenge ciphertext C^* and he should not be able to derive another ciphertext C such that its decapsulated key K is non-trivially related to challenge key K^*. Here, we introduce three NM definitions of KEM, and show that the three definitions are equivalent.

It is easily obtained from one of the definitions of NM that NM-CCA2 KEM is equivalent to IND-CCA2 KEM. That is, we can now recognize that Shoup's definition, IND-CCA2, for KEM is as feasible as NM-CCA2, whereas NM itself is stronger than IND in the definition.

In addition, this thesis investigates other stronger definitions, i.e., the UC security for KEM and DEM. The UC framework was introduced by Canetti [13] and it guarantees very strong security, i.e., preserves stand-alone security in any type of composition with other primitives and protocols.

Although the UC security for KEM and DEM, as the ideal functionalities of KEM and KEM-DEM, has been defined and investigated in [41, 42], this thesis modifies the definition, security proof, and description as described hereafter.

1. In the previous definition of the functionality of KEM-DEM, only a single shared key was available in the DEM phase. This thesis modifies the functionality of KEM-DEM to remove this restriction so that a single copy of the functionality of KEM-DEM accepts multiple shared keys in the DEM phase.

2. Another problem in [41, 42] is the proof that UC KEM equals NM-CCA2, i.e., IND-CCA2), KEM. The proof was based on a previous

definition of NM which is, as mentioned above, only available for a few types of KEM schemes. This thesis corrects the proof of the equality between UC KEM and IND-CCA2 KEM, in which it is directly proven without using any NM definition (it is equivalent to the proof through the new NM definition).

3. This thesis follows the new framework of UC that was totally revised by Canetti in 2005 [13], while [41] and [42] are based on the original one in 2001. The equivalence between UC DEM and IND-P2-C2 DEM is also proven (through no NM) in this thesis.

Finally, we studied the relationship among three cryptographic channels, SC, AC, and DIC using UC with the trendy framework probabilistic input/output automata (PIOA). The SC was based on combining KEM and DEM. The relationship of the three cryptographic channels was investigated by Okamoto. [46]. He showed that the three cryptographic channels are reducible to each other, but did not consider communication schedules clearly as well as composable security. This thesis refines the relationship of the three channels in light of communication schedules and composable security. We model parties using the task-PIOA to treat communication schedules, and adopt the UC framework by Canetti to treat composable security. We show that a class of anonymous channels, two-anonymous channels (2AC), and DIC are reducible to each other under some types of schedules and that DIC and SC are reducible to each other under some types of schedules in the UC framework with the PIOA model.

1.2 Structure of the Thesis

This thesis comprises eight chapters including this chapter as the introduction.

Chapter 2 introduces KEM and DEM as ISO standards. A protocol that combines KEM and DEM, called hybrid public key encryption (HPKE), is also introduced in this chapter. First, we introduce the basic cryptographic protocol, public key encryption (PKE), that is based on KEM, DEM, and HPKE. Second, the mechanisms and security notions of KEM and DEM are described and the attack types of these mechanisms are explained. Finally, we describe the mechanism of HPKE.

Chapter 3 introduces a UC framework that was introduced by Canetti [13]. This framework is based on interactive turing machines (ITMs) and solving the difficulty in composability of several protocols. This chapter describes the main notion of UC, the security theorem, and hybrid theorem. First, these theorems are considered based on two basic models (or worlds), one is a real life model (or real world) and the other is an ideal process model (or ideal world). These two models enable us to consider security from a real protocol to an ideal function. The real life model consists of three elements: parties, adversary, and protocol. The ideal process world consists of three elements: dummy parties, simulator, and ideal functionality. The party that inputs some messages into the parties, called the environment, is also an important party. This chapter also explains adversarial models: non-adaptive adversary, static corruption adversary, and adaptive adversary. These are important when we consider the security levels of real protocols. Finally, we describe the UC security. This security is described as hybrid model of UC. The hybrid model is a specification of the real life model that is assisted by some ideal functionalities. We consider an unlimited number of copies of the ideal functionalities.

Chapter 4 introduces the framework for PIOA. This chapter also introduces the task-PIOA. The concept of task-PIOA is simple and essential to consider the composability regarding PIOA. In particular, it is important to obtain results considering concurrency, asynchronous schedule, and synchronous schedule for a security model. The UC model is inefficient with respect to the schedule because it is based on ITMs. Therefore, to obtain equivalence results among the three channels, we must consider the master schedule among parties. We explain the basic definitions and theorems or corollaries of PIOA and task-PIOA in this chapter.

Chapter 5 shows the results of semantic security for KEM and DEM. Moreover, this chapter presents the definitions of three NMs for KEM and shows the proof of equivalence among the three NMs for KEM. We also show the main theorem for the security of KEM and the proof of equivalence, IND-CCA2 KEM is equivalent to NM-CCA2 KEM. In addition, we introduce IND and NM for DEM and prove the equivalence between two notions of DEM from [41, 42] to make the following chapter easier to understand.

Chapter 6 presents the result for UC KEM and UC DEM. We define the functionality of KEM in the UC model and show the main theorem for UC KEM and the proof of equivalence. UC KEM is equivalent to IND-CCA2 KEM (IND-CCA2 KEM is the highest security level for KEM). We also define the functionality of DEM in the UC model and show the main theorem for UC DEM and the proof of equivalence. UC DEM is equivalent to IND-P2-C2 DEM (IND-P2-C2 DEM is the highest security level for DEM).

Chapter 7 presents three results. Three cryptographic channels are reducible to each other in the UC framework with task-PIOA. First, we explain the three basic channels: SC, 2AC, and DIC. Next, we define the task with respect to task-PIOA. More precisely, we define the codes for the functionalities of the three channels and prove the reductions among the three cryptographic channels in the UC framework with task-PIOA. Finally, the reduction proofs are presented in this chapter considering the master schedules of task-PIOA.

Chapter 8 concludes the thesis and summarizes the results obtained through this research.

Chapter 2

Hybrid Public Key Encryption

2.1 Preliminaries

Based on standard convention, we use four notations, negligible function, probabilistic algorithm, experiment and vector as follows where $\mathtt{N}$ is the set of natural numbers, $\mathtt{R}$ is the set of real numbers and $\perp$ denotes a null string.

1. **Negligible function:**
   Let k be a security parameter. Function $f : \mathtt{N} \rightarrow \mathtt{R}$ is negligible in k, if for every constant $c > 0$, there exists integer k_c such that $f(k) < k^{-c}$ for all $k > k_c$. Hereafter, we often use $f < \epsilon(k)$ to mean that f is negligible in k. On the other hand, we use $f > \mu(k)$ to mean that f is not negligible in k, i.e., function $f : \mathtt{N} \rightarrow \mathtt{R}$ is not negligible in k, if there exists constant $c > 0$ such that for every integer k_c, there exists $k > k_c$ such that $f(k) > k^{-c}$.

2. **Probabilistic algorithm:**
   Let $A(x_1, x_2, \cdots ; r)$ denote the results of executing A that takes as inputs $x_1, x_2, \cdots$ and coins r. Let y be the output by executing $A(x_1, x_2, \cdots ; r)$, that is $A(x_1, x_2, \cdots ; r) = y$.

3. **Experiment:**
   Let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of selecting r at random

and letting y equal the output of $A(x_1, x_2, \cdots; r)$. If S is a finite set, then $x \leftarrow S$ denotes the experiment of assigning to x an element uniformly chosen from S. If α is neither an algorithm nor a set, then $x \leftarrow \alpha$ indicates that we assign α to x. When A is a probabilistic machine or algorithm, $A(x)$ denotes the random variable of the output of A with regard to input x. Term $y \overset{\mathsf{R}}{\leftarrow} A(x)$ denotes that y is randomly selected from $A(x)$ according to its distribution. When A is a set, $y \overset{\mathsf{U}}{\leftarrow} A$ denotes that y is uniformly selected from A. When A is a value, $y \leftarrow A$ denotes that y is set as A.

4. **Vector:**

   - We describe vectors in boldface as $\boldsymbol{x}$ and denote the number of components in $\boldsymbol{x}$ by $|\boldsymbol{x}|$, and the i-th component by $\boldsymbol{x}[i]$ so that $\boldsymbol{x}$ = $(\boldsymbol{x}[1], \cdots, \boldsymbol{x}[|\boldsymbol{x}|])$.
   - We denote a component of a vector as $x \in \boldsymbol{x}$ or $x \notin \boldsymbol{x}$, which respectively means that x is in or is not in the set $\{\boldsymbol{x}[i] : 1 \leq i \leq |\boldsymbol{x}|\}$. From the above, we can simply describe $\boldsymbol{x} \leftarrow A(\boldsymbol{y})$ as the shorthand form of $1 \leq i \leq |\,\boldsymbol{y}\,|$ do $\boldsymbol{x}[i] \leftarrow A(\boldsymbol{y}[i])$.
   - Instead of describing $R(x_1, \cdots, x_t)$, we describe $R(x, \boldsymbol{x})$, which means that the first argument is special and the rest are collectively given as vector $\boldsymbol{x}$ with $|\boldsymbol{x}| = t - 1$. Here, we consider relations of amity t where t is a polynomial in security parameter k.

2.2 Public Key Encryption

The concept of public key encryption (PKE) is simple and compendious, but has far-reaching consequences. This encryption method or scheme has been known since the 1970s. This is called public key (or asymmetric key) scheme / encryption because the decryption key is different from the encryption key. Furthermore, it is infeasible to find the decryption key from the encryption key and the message from the ciphertext, respectively. This method is constructed from three algorithms $\mathsf{PKE} = (K, E, D)$ where K is a key generation algorithm, E is an encryption algorithm, and D is a decryption algorithm. Let (PK, SK) be a pair comprising a public key and secret key generated by K. The algorithms E and PK are open to us, but it is computationally infeasible, given random ciphertext $c \in C$, to find message $m \in M$ such that $E(m) = c$. This property implies that given public key PK it is difficult to determine the corresponding secret key SK. The public key PK is different from secret key SK in public key encryption while it is the same key in symmetric key encryption.

2.3 Key Encapsulation Mechanism

A KEM was proposed as an ISO standard by Shoup. A KEM, Σ, is given by three algorithms $\Sigma = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ where

1. $\mathcal{G}$, the key generation algorithm, is a probabilistic polynomial time (PPT) algorithm that takes security parameter $k \in \mathtt{N}$ (provided in unary) and returns pair (pk, sk) comprising a matching public key and secret key.

2. $\mathcal{E}$, the key encryption algorithm, is a PPT algorithm that takes as input public key pk and outputs key/ciphertext pair (K^*, C^*).

3. $\mathcal{D}$, the decryption algorithm, is a deterministic polynomial time algorithm that takes as input secret key sk and ciphertext C^*, and outputs key K^* or $\perp$ ($\perp$ implies that the ciphertext is invalid).

We require that for all (pk, sk) output by key generation algorithm $\mathcal{G}$ and for all (K^*, C^*) output by key encryption algorithm $\mathcal{E}(pk)$, $\mathcal{D}(sk, C^*) = K^*$ holds. Here, the length of the key, $|K^*|$, is specified by $l(k)$, where k is the security parameter.

2.3.1 KEM Attack Types

The three attack types of KEM are CPA, CCA1 and CCA2, where:

1. **CPA**, Chosen Plaintext Attack, is an attack type that an adversary is allowed to access to only encryption oracle but not decryption oracle.

2. **CCA1**, Chosen Ciphertext Attack, is an attack type that an adversary is allowed to access to both encryption and decryption oracle. However the adversary cannot access to decryption oracle after getting target ciphertext.

3. **CCA2**, Adaptive Chosen Ciphertext Attack, is an attack type that an adversary is allowed to access to both encryption and decryption oracle even after the adversary gets target ciphertext.

Note that the adversary cannot decrypt the target ciphertext in the case of CCA2. Each attack type is defined by considering an adversary that can access the encryption oracle and decryption oracle. The encryption oracle takes plaintexts as input and returns ciphertexts that are the encryption of input plaintexts. On the other hand, the decryption oracle takes ciphertexts as input and returns plaintexts that are the decryption of input ciphertexts. Adversaries are able to attack the target ciphertext under the conditions of the above attack types. However, the adversary cannot access the decryption oracle with the target ciphertext.

2.3.2 IND for KEM

The indistinguishability (IND) of KEM was defined by Shoup [55]. We use "IND-ATK-KEM" to describe the security notion of indistinguishability for KEM against $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$. "IND-KEM" is used to focus on the indistinguishability of KEM without regard to attack type. If it is clear from the context that IND-ATK-KEM (and IND-KEM) is used for KEM, we will call it IND-ATK (and IND) for simplicity.

To clarify the IND of PKE, we may use IND-ATK-PKE and IND-PKE.

$$\mathtt{Adv}^{\mathsf{IND\text{-}ATK}}_{A,\Sigma}(k) \equiv \Pr[\mathtt{Expt}^{\mathsf{IND\text{-}ATK}}_{A,\Sigma}(k) = 1] - \frac{1}{2},$$

where $\mathtt{Expt}^{\mathsf{IND\text{-}ATK}}_{A,\Sigma}(k)$:

$$(pk, sk) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^k);\ s \overset{\mathsf{R}}{\leftarrow} A_1^{O_1}(pk);$$
$$(K^*, C^*) \overset{\mathsf{R}}{\leftarrow} \mathcal{E}(pk);\ R \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)};\ b \overset{\mathsf{U}}{\leftarrow} \{0,1\};$$
$$X \leftarrow \begin{cases} K^*, \text{ if } \ b = 0 \\ R, \text{ if } \ b = 1 \end{cases}$$
$$g \overset{\mathsf{R}}{\leftarrow} A_2^{O_2}(s, X, C^*); \text{return 1, iff } \ g = b$$

and
If ATK = CPA, then $O_1 = \perp$ and $O_2 = \perp$.
If ATK = CCA1, then $O_1 = \mathcal{D}(sk, \cdot)$ and $O_2 = \perp$.
If ATK = CCA2, then $O_1 = \mathcal{D}(sk, \cdot)$ and $O_2 = \mathcal{D}(sk, \cdot)$.

Figure 2.1: Advantage of IND-ATK-KEM

Definition 1. *Let* Σ *be* KEM, $A = (A_1, A_2)$ *be an adversary, and* $k \in \mathbb{N}$ *be a security parameter. For* ATK $\in$ {CPA, CCA1, CCA2}, $\mathtt{Adv}^{\mathsf{IND\text{-}ATK}}_{A,\Sigma}(k)$ *is defined in Fig. 2.1. We say that* Σ *is IND-ATK-KEM, if for any adversary* $A \in \mathcal{P}$, $\mathtt{Adv}^{\mathsf{IND\text{-}ATK}}_{A,\Sigma}(k)$ *is negligible in* k *where* ATK $\in$ {CPA, CCA1, CCA2} *and* $\mathcal{P}$ *denote a class of polynomial-time bounded machines.*

2.4 Data Encapsulation Mechanism

A DEM was also proposed as an ISO standard by Shoup. A DEM, Σ', is given by two algorithms $\Sigma' = (\mathcal{E}', \mathcal{D}')$ where

1. $\mathcal{E}'$, the data encryption algorithm, is a PPT algorithm that takes as input secret key K (K is shared by KEM) and plaintext M, and outputs ciphertext C.

2. $\mathcal{D}'$, the data decryption algorithm, is a deterministic polynomial time

algorithm that takes as input secret key K and ciphertext C, and outputs plaintext M or $\perp$ ($\perp$ implies that the ciphertext is invalid).

It is required that for all C output by data encryption algorithm $\mathcal{E}'(K, M)$, $\mathcal{D}'(K, C) = M$ holds (”soundness”). Here, the length of the key, $|K|$, is specified by $l(k)$ where k is the security parameter.

2.4.1 DEM Attack Types

From the standard notion of the attack type, we consider the following nine DEM attack types: PX-CY (X$\in \{0, 1, 2\}$ and Y$\in \{0, 1, 2\}$), i.e., P0-C0, P1-C0, P2-C0, P0-C1, P1-C1, P2-C1, P0-C2, P1-C2, and P2-C2.

1. PX (X$\in \{0, 1, 2\}$) denotes access to the encryption oracle. P0 means that the adversary does not have access to the encryption oracle. P1 means “Chosen Plaintext Attacks” where the adversary is allowed to access the encryption oracle, but cannot access the encryption oracle after obtaining the target ciphertext. P2 means “Adaptive Chosen Plaintext Attacks” where the adversary is allowed to access the encryption oracle, even after it obtains the target ciphertext.
2. CY (Y$\in \{0, 1, 2\}$) denotes access to the decryption oracle. C0 means that the adversary does not have access to the decryption oracle. C1 means “Chosen Ciphertext Attacks” where the adversary is allowed to access the decryption oracle, but cannot access the decryption oracle after obtaining the target ciphertext. C2 means “Adaptive Chosen Ciphertext Attacks” where an adversary is allowed to access the decryption oracle after it obtains the target ciphertext, but the adversary cannot decrypt the target ciphertext in the case of C2 for all PX.

2.5 Hybrid Public-Key Encryption

Using a canonical way to compose KEM and DEM, we obtain a hybrid public key encryption scheme.

First, let $\Sigma = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be KEM and let $\Sigma' = (\mathcal{E}', \mathcal{D}')$ be DEM. Let *KEM.KeyLen* be the length of the output key of KEM and *DEM.KeyLen* be the length of the output key of DEM. Second, to compose these two mechanisms, we require that they are compatible in the sense that *KEM.KeyLen* = *DEM.KeyLen*.

We state a hybrid public key encryption scheme $\mathsf{H\text{-}PKE} = \mathsf{H\text{-}PKE}_{\mathrm{KEM,DEM}}$ in terms of KEM and DEM as given hereafter.

1. **Key generation algorithm**
   The key generation algorithm in H-PKE is the same as KEM.KeyGen(). The obtained key, (pk, sk), by KEM.KeyGen() represents a public key and secret key, respectively.

2. **Encryption algorithm**
   First, the encryption algorithm in H-PKE executes $\mathcal{E}(pk)$ to generate ciphertext C_0 and shared key K. Second, it encrypts message M to C_1 under K using $\mathcal{E}'(K, M)$. Third, it outputs ciphertext $C = C_0 \parallel C_1$. Any of these steps may fail, in which case the encryption algorithm in H-PKE also fails.

3. **Decryption algorithm**
   First, the decryption algorithm in H-PKE parses C as $C = C_0 \parallel C_1$ using the prefix-freeness property [*] of the ciphertexts. Second, it decrypts C_0 to shared key K under sk using $\mathcal{D}(sk, C_0)$. Third, it decrypts C_1 to message M using $\mathcal{D}'(K, C_1)$. Any of these steps may fail, in which case the decryption algorithm in H-PKE also fails.

Shoup defined a hybrid public key encryption scheme as the above H-PKE for an ISO standard (see [55] for more details).

[*] The set of all possible outputs of the encryption algorithm should be a subset of some easy to recognize prefix-free language — language L is prefix free if for any two x, y ∈ L, x is not a proper prefix of y. The prefix-freeness property is needed so that we can parse byte strings from left to right, and efficiently strip off a ciphertext. Note that if all ciphertexts have the same length, then the prefix-freeness property is trivially satisfied.

Chapter 3

Universal Composability

The notion of UC was formalized by Canetti [13]. Informally, we redescribe the UC framework.

3.1 Overview

UC framework treats a protocol as a system of Interactive Turing Machines (ITMs) and assume that all ITMs run in probabilistic polynomial time (PPT). So, each party runs the program in the ITM system.

In the UC framework, there are input, output, and communication tapes. The input and output tapes represent the input and output that are received from and given to other systems (involving some programs) running on the same machine, respectively. On the other hand, the messages in the communication tapes are sent to and received from the participants in the system. In addition, an adversary is also modeled as an ITM. The merits of this framework are itemized below.

- In the sense of UC security, we can securely achieve (UC-realize) a Byzantine system by combining some UC secure briefest units.

- In clear and simple terms, we can understand and construct a complex system.

The following describes how the Byzantine system is UC-realized from some small systems/protocols. First, we assume that small systems are UC secure. By considering "ideal functionality", we assume the systems can be treated as functionality. Second, we must say that the constructed (Byzantine) system is UC secure by using small ideal functionality. The UC hybrid model assures that the ideal functionalities construct a secure combined system. Here, we refer to the basic notion of this framework as indicated hereafter.

3.2 Security Framework

Protocol π is executed in the following three steps:

1. A real life model (or real world) is formalized to carry out a task of protocol π. The adversary or the environment executes π with some parties in this real life model.

2. An ideal process model (or ideal world) is formalized to idealize the executions in the real life model. In this model, the parties communicate with each other through an "ideal functionality." The ideal functionality is a functionality to achieve the desired functionality of the task in the real life model and it is essentially an incorruptible trusted party.

3. The environment is activated to distinguish computationally between the real life model and ideal process model.

Informally, we say that protocol π UC-realizes an ideal functionality if the environment cannot distinguish the execution of the real life process from the ideal process model. More details regarding the real life world and the ideal process world are given below.

3.2.1 Real Life Model

In the real life model, the real world has parties P_i, environment Env, and an adversary Adv as participants. Parties P_i execute protocol π in conjunction with Adv and environment Env based on the input z from Env. All participants have security parameter k and they are able to perform the actions below.

- Either P_i, Env, or Adv is activated to execute π by inputting a message as their respective input or in their respective inpucoming communication tapes. The input or input communication tape is modeled as an ITM as well. From the input tape or incoming communication tapes the activated participant reads some information, executes its program, and outputs information on its output tape or outgoing communication tapes. In addition, the environment can read the output tapes of the parties P_i, and write some information on the input tapes of the parties P_i.

- Adv is able to read messages on the outgoing communication tapes and can deliver these messages to the incoming message tapes of the recipient parties. Note that only messages generated by P_i can be delivered.

- Adv is also able to corrupt parties P_i according to four types of corruption.

  - An adaptive adversary may corrupt the parties P_i while the protocol is executed.
  - A non-adaptive adversary can corrupt party P_i before any party P_i is activated.
  - An active adversary can establish complete control over the behavior of the corrupted parties P_i.
  - A passive adversary can obtain only the internal information of the corrupted parties P_i. In the case of a passive adversary, cor-

rupted parties P_i can continue to follow their prescribed protocol.

Formally, the notion of security in the real life model is denoted as follows. Let $Real_{\pi,Adv,Z}(k,z,\boldsymbol{r})$ denote the output of environment Env when interacting with adversary Adv and parties P_1, ..., P_n executing protocol π on security parameter k, input z and random input $\boldsymbol{r}$ = r_{Env}, r_{Adv}, r_1 ... r_n be as described above (z and r_{Env} for Env; r_{Adv} for Adv; r_i for party P_i). Let $Real_{\pi,Adv,Env}(k,z)$ denote the random variable describing $Real_{\pi,Adv,Env}(k,z,\boldsymbol{r})$ when $\boldsymbol{r}$ is uniformly chosen. Let $Real_{\pi,Adv,Env}$ denote the ensemble $\{Real_{\pi,Adv,Env}(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$.

3.2.2 Ideal Functionality Model

In the ideal process model, there is a simulator, Sim, and ideal functionality $\mathcal{F}$. Sim and $\mathcal{F}$ proceed with Env and dummy party P_i in the ideal process world as follows:

- Env is activated with its input z, and activates Sim and P_i.
- Sim shall simulate the protocol π in the real world faithfully. Sim proceeds to deceive environment Env that has interaction, real life world or ideal world. Furthermore, Sim simulates protocol π with ideal functionality.
- $\mathcal{F}$ communicates with some dummy parties P_i and Sim interactively in the PPT. In other words, once $\mathcal{F}$ is activated, it reads the information in its incoming communication tape, and outputs to dummy parties P_i or sends messages to simulator Sim.

Formally, the notion of security in the ideal process model is denoted as follows. Let $Ideal_{\mathcal{F},Sim,Env}(k,z,\boldsymbol{r})$ denote the output of environment Env after interacting in the ideal process with adversary Sim and ideal functionality $\mathcal{F}$, on security parameter k, input z, and random input $\boldsymbol{r}$ = r_{Env}, r_{Sim}, $r_{\mathcal{F}}$ as described above (z and r_{Env} for Env, r_{Sim} for Sim;

$r_{\mathcal{F}}$ for $\mathcal{F}$). Let $Ideal_{\mathcal{F},Sim,Env}(k,z)$ denote the random variable describing $Ideal_{\mathcal{F},Sim,Env}(k,z,\boldsymbol{r})$ when $\boldsymbol{r}$ is uniformly chosen. Let $Ideal_{\mathcal{F},Sim,Env}$ denote the ensemble $\{Ideal_{\mathcal{F},Sim,Env}(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$.

3.2.3 UC Security

We say that protocol π UC-realizes ideal functionality $\mathcal{F}$ if for any real life model adversary, *Adv*, there exists an ideal process model adversary, *Sim*, such that no environment, *Env*, on any input, can tell with non-negligible probability whether it is interacting with *Adv* and parties executing P_i in the real life model, or it is interacting with *Sim* and $\mathcal{F}$ in the ideal process model. This means that, from the viewpoint of the environment, executing protocol π is just as good as interacting with an ideal process for $\mathcal{F}$. Formally, Environment Security(ES) is denoted as follows.

Let $\mathcal{F}$ be an ideal functionality and let π be a protocol. We say that π UC-realizes $\mathcal{F}$ if for any adversary *Adv* there exists an ideal process model adversary, *Sim*, such that for any environment *Env* we have :

$$Ideal_{\mathcal{F},Sim,Env} \approx Real_{\pi,Adv,Env}.$$

3.2.4 Non-Adaptive / Adaptive Adversary

Two adversary attack types are modeled in the UC framework. The first is a non-adaptive adversary in which adversary *Adv* cannot collapse and withdraw the parties in the running of the real life protocol. Another is an adaptive adversary that can collapse and withdraw the parties in the execution of the real life protocol. For parties Init and Rec in protocol π, there are four statuses.

Status 1 : No party is corrupted by *Adv*

All of the parties in protocol π are safe because *Adv* collapses no party. That is, this status shows that protocol π is executed securely between Init and Rec because *Adv* can obtain no information except with the outputs of the parties and the information forwarded from *Env*.

Status 2 : Only Init **is corrupted by** *Adv*

Adversary *Adv* collapses only Init upon the input by *Env*. This status shows that protocol π may not be executed securely between Init and Rec because *Adv* can obtain any information from the corrupted party Init.

Status 3 : Only Rec **is corrupted by** *Adv*

The adversary collapses only Rec upon the input by *Env*. This status shows that protocol π may not be executed securely between Init and Rec because *Adv* can obtain any information from the corrupted party Rec.

Status 4 : Both parties are corrupted by *Adv*

The adversary collapses both Init and Rec upon the input by *Env*. This status also shows that protocol π is not executed securely between Init and Rec because *Adv* can obtain any information from corrupted parties Init and Rec, and the information forwarded from *Env*.

- **Non-Adaptive Adversary** The non-adaptive adversary never transits from the first status to another status when executing protocol π. Therefore, the first status, which is fixed by *Env* before π begins, continues to the end of the protocol π.

- **Adaptive Adversary** The adaptive adversary can collapse and withdraw the parties in the execution of the real life protocol although the non-adaptive adversary cannot. *Adv* can change from the first status to another status when *Env* activates *Adv* to collapse or withdraw the target party. The status may transit to any of the above-mentioned four statuses anytime.

3.2.5 Hybrid Model

The UC framework formalizes the hybrid model as described hereafter. The hybrid model is specified as a real life model that is assisted by some ideal functionality, $\mathcal{F}$ (in short, the $\mathcal{F}$-hybrid model). We consider an unlimited number of copies of ideal functionality $\mathcal{F}$. The copies of $\mathcal{F}$ are differentiated using their session ID, *sid*. The parties and the adversary may send messages to and receive messages from each copy of $\mathcal{F}$ in each activation. If they send messages to each copy of $\mathcal{F}$ using *sid*, they write information on the incoming communication tape of that copy. On the other hand, if $\mathcal{F}$ sends outgoing messages, adversary H delivers the messages but is barred access to the contents of that message.

Let $Hyb^{\mathcal{F}}_{\pi,Adv,Env}(k,z)$ denote a random variable describing the output of environment *Env* on input z, after *Env* interacts in the $\mathcal{F}$-hybrid model with protocol π and adversary *Adv* (We stress that here π is a hybrid protocol with ideal functionality $\mathcal{F}$). Let $Hyb^{\mathcal{F}}_{\pi,Adv,Env}$ denote the distribution ensemble $\{Hyb^{\mathcal{F}}_{\pi,Adv,Env}(k,z)\}_{k\in N,z\in\{0,1\}^*}$.

3.2.6 UC Hybrid Security

The universal composition theorem is hereafter. Let $\mathcal{F}$ and $\mathcal{G}$ be ideal functionalities. Let π be a protocol in the $\mathcal{F}$-hybrid model, and let π^{ρ} be a protocol that UC-realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model. Then for any adversary $Adv_{\mathcal{G}}$ there exists adversary $Adv_{\mathcal{F}}$ such that for any environment *Env* we have

$$Hyb^{\mathcal{F}}_{\pi,Adv_{\mathcal{F}},Env} \approx Hyb^{\mathcal{G}}_{\pi^{\rho},Adv_{\mathcal{G}},Env}.$$

Chapter 4

(Task) Probabilistic I/O Automata

4.1 Introduction

This chapter introduces PIOA and task-PIOA that can model and verify a security protocol. Task-PIOA enables us to consider the protocols as the automaton of tasks defined in the task-PIOA settings. We must consider composable security using the UC framework, but the framework is based on ITMs. Therefore, we must compensate for the lack of power due to the ITM construction because ITM is sequencial model not the concurrent model. More specifically, we need a schedule property when we consider reducing the SCs, ACs, and DICs. As a result, we focus on the UC with task-PIOA considering these three cryptographic channels.

4.2 Preliminaries

This section introduces the basic notions of mathematics on task-PIOA from [15, 16, 17, 18].

Probabilistic measure Let X and μ be a set and a discrete probability measure, respectively. μ is the discrete probability measure on set X. A σ-field over set X is set $F \subseteq 2^X$. F contains the empty set and F can be closed. If F is a σ-field over X, then the pair of (X,F) is a measurable space. Let

μ be function $F \rightarrow [0, \infty]$ and μ be countably additive. The other details are described in [18].

Support Support of probability measure μ is measurable set C such that $\mu(C) = 1$. If probability measure μ is a discrete probability measure, we denote this by supp(μ) where the support of probability measure μ is the set of elements that have non-zero measure.

Apply Function $apply(\mu,\rho)$ takes discrete probability measure μ on finite execution fragments (see 4.3) and task schedule ρ (see 4.5.1), and returns a probability measure on execution fragments. Note that this function satisfies apply(μ, λ) = λ (λ is the empty sequence).

4.3 Probabilistic I/O Automaton (PIOA)

The PIOA framework was introduced by Segala in [49, 51, 52] to analyze probabilistic distributed algorithms. The PIOA framework treats the probabilistic and nondeterministic choices in the notion. To analyze the level of cryptographic security and resolve the problem of concurrency, we essentially need the notion of nondeterminism.

Definition 2. [PIOA:] *Let P be PIOA that is a tuple of* $(Q, \overline{q}, I, O, H, D)$ *[18] as follows:*

- *Q is a countable set of states.*
- *$\overline{q}$ describes a start state and satisfies $\overline{q} \in Q$.*
- *I is a countable set of input actions.*
- *O is a countable set of output actions.*
- *H is a countable set of internal actions.*
- *D is a transition relation satisfying $D \subseteq Q \times (I \cup O \cup H) \times Disc(Q)$, where $Disc(Q)$ is the set of discrete probability measures on Q.*

The following sets are also defined:

- *A is $I \cup O \cup H$, this represents the set of actions.*

- *E denotes $I \cup O$, this represents the set of external actions.*

- *L denotes $O \cup H$, this represents the set of locally controlled actions.*

Note that the sets I, O, and H are pairwise disjoint sets, respectively. If $I = \emptyset$, then P is closed.

We say an action a is enabled *in state q if $(q, a, \mu) \in D$ for some μ. We assume that P satisfies two properties, input enabling and transition determinism. The input enabling means that for every $q \in Q$ and $a \in I$, a is enabled in q. The transition determinism means that for every $q \in Q$ and $q \in A$, there is at most one $\mu \in Disc(Q)$ such that $(q, a, \mu) \in D$.*

Let q_i and a_i for $i \in \{0, 1, 2, \cdots\}$ be states and actions, respectively. We consider that an execution fragment of PIOA P is the following infinite or finite sequence $\alpha = q_0 a_1 q_1 a_2 \ldots$. If sequence α is a finite sequence, the last state of α is denoted by $\mathrm{lst}(\alpha)$. If α is a finite sequence with $\mathrm{lst}(\alpha) = q_{i+1}$, for each (q_i, a_{i+1}, q_{i+1}) there exists a transition $(q_i, a_{i+1}, \mu) \in D$ with $q_{i+1} \in \mathrm{supp}(\mu)$, where $\mathrm{supp}(\mu)$ is a support of μ. Here, we use the term of "$Frags(P)$" (resp., "$Frags^*(P)$") to denote the set of all (resp., all finite) execution fragments of P. We then use the term of "$Execs(P)$" (resp., "$Execs^*(P)$") to denote the set of all (resp., all finite) executions of P. If there exists execution fragment α of P, we denote the input and output (external actions) sequence obtained from α as $trace(\alpha)$.

4.4 Task-PIOA

The perfect information schedules of PIOA are very powerful in analyzing the security of protocols. The schedule based on the all information included the local information that parties have. The perfect information schedules are open to all participants. However, the adversary cannot access to the local information in the case of non-corrupted case when we analize the security of protocols. Therefore, the local information should be hided against the perfect information schedules or adversary to execute tasks nondeterministically.

When an adversary corrupts a party, the secret information that the party retain in the internal process is reveal to the adversary. For this problem, we must apply the non-adaptive task schedule mechanism. This mechanism is defined in task PIOA. In simple terms, the task is used as units of scheduling.

Definition 3. [Task-PIOA:] *Task-PIOA T is defined to be pair (P, R) where*

- P is a PIOA $(Q, \bar{q}, I, O, H, D)$,
- R is an equivalence relation on $L = O \cup H$.

The task is an equivalence class of R. We say that task t is enabled in state q if some $a \in t$ is enabled in q. We require that every task-PIOA T satisfies *action determinism* property. The *action determinism* means that for every state $q \in Q$ and every task $t \in R$, there is at most one action $a \in t$ that is enabled in q.

The tasks can be used to resolve all nondeterminisms using action determinism and transition determinism for PIOAs. In other words, subsequent action is specified with the given state and by specifying a task.

4.5 Schedules

In this thesis, we formally model parties $P_1, \cdots, P_n$ in a protocol using task-PIOA $T_1, \cdots, T_n$. Each P_i executes its task according to the following task schedule, local schedule, and master schedule.

4.5.1 Task Schedule

We refer to the notion of "task schedule," which chooses the next task to perform. For a closed task-PIOA, i.e., one with no input actions, a task schedule resolves all nondeterminism due to the next-action determinism property of task-PIOAs and the next-transition determinism property of general PIOAs.

Definition 4. [Task Schedule:] *Let $T = (P, R)$ be a closed task-PIOA where $P = (Q, \overline{q}, I, O, H, D)$. A task schedule for T is defined to be a finite or infinite sequence $\rho = t_1 t_2 \dots$ of tasks in R.*

Here, for a task schedule ρ, we define the trace distribution $tdist(\rho)$. The trace distribution, $tdist(\rho)$, is the image measure of $apply(\delta(\overline{q}), \rho)$ where $\delta(\overline{q})$ is the Dirac measure on the start state $\overline{q}$. We define the set of trace distributions of T, $tdists(T)$, to be $\{tdist(\rho) | \rho$ is a task schedule for $T\}$.

4.5.2 Local Schedule

A second schedule is introduced to the local schedule to resolve nondeterminisms. The local schedule resolves the nondeterminisms within the system components based on local information. This schedule differs from the task schedule in the point of no action determinism assumption. The local schedule for task-PIOA T is defined to be as follows.

Definition 5. [Local Schedule:] *Let $T = (P, R)$ and s be a closed task-PIOA for party P and local information, respectively. A local schedule, $\omega(s)$, for T is defined to be a finite or infinite sequence of tasks, $t_1, t_2, \cdots$, i.e., $\omega(s) = t_1, t_2, \cdots$. $\omega(s)$ specifies the execution order of the tasks in R with s (We often omit from the specification the explicit description of $\omega(s)$ in the specification of a task-PIOA if ω and s where s includes λ, are trivial).*

4.5.3 Master Schedule

Definition 6. [Master Schedule:] *A master schedule, M, is defined to be a finite or infinite sequence of party identifiers, $i_1, i_2, \cdots$, i.e., $M = i_1, i_2, \cdots$. M globally specifies the execution order of tasks in a protocol of $(P_1, \cdots, P_n)$ while preserving the local scheduling of all parties.*

For example, let ρ_i of party i be $t_{i1}, t_{i2}, \cdots$ $(i = 1, 2, 3)$, and $M = 1, 2, 2, 2, 3, 1, 1, 3$. Then the global execution order of tasks is $t_{11}, t_{21}, t_{22}, t_{23}, t_{31}, t_{12}, t_{13}, t_{32}$.

The master schedule is not under the control of an adversary although the local schedule is under the control of an adversary. In other words, the adversary cannot intervene on the master schedule, but he can encumber a local schedule. This construction can be generalized from a single execution fragment α to a discrete probability measure ϵ on execution fragments. See [15, 16] for details.

In [15, 16], a master schedule is defined to resolve nondeterminism. The master schedule controls local schedules in a large system nondeterministically or deterministically. A local schedule helps the master schedule from within the system components and uses only local information. One of the ways to realize the task schedule is using local local schedule and (/or) master schedule.

4.5.4 Operations

The other operations for task-PIOAs are defined in [18]. The formal definitions and theorems are available in that paper. Here, we briefly describe operations, composition, and hiding operations hereafter.

Definition 7. [Composition Operation:] *For two compatible task-PIOAs $T_1 = (P_1, R_1)$ and $T_2 = (P_2, R_2)$, their composition $T_1 \parallel T_2$ is defined to be task-PIOA $(P_1 \parallel P_2, R_1 \cup R_2)$.*

Definition 8. [Hiding Operation:] *For task-PIOA $T = (P, R)$ and set $S \subseteq O$ of output actions, $hide(T, S)$ is defined to be $(hide(P, S), R)$ where $hide(P, S)$ yields $(Q, \bar{q}, I, O \setminus S, H \cup S, D)$ for $P = (Q, \bar{q}, I, O, H, D)$.*

Here, we introduce the *full* operation as described hereafter. If $T_1 = (P_1, R_1)$ and $T_2 = (P_2, R_2)$ are two task-PIOAs, and if $c : (R_1^* \times R_1) \to R_2^*$, then we define $full(c) : R_1^* \to R_2^*$ recursively, as

- $full(c)(\lambda) = \lambda$
- $full(c)(\rho T) = full(c)(\rho)c(\rho, T)$.

4.5.5 Security Definitions

We describe “comparable” and “environment” definitions as follows.

Definition 9. *Two task-PIOAs T_1 and T_2 are comparable if $I_1 = I_2$ and $O_1 = O_2$, that is, if they have the same input actions and the same output actions.*

Definition 10. *If T and E are task-PIOAs, then E is said to be an environment for T if T and E are compatible, and $T \parallel E$ is closed.*

We now introduce implementation, the basic definition of security for task-PIOA as described in [18].

Definition 11. [Implementation:] *Suppose T_1 and T_2 are two comparable task-PIOAs. We say that $T_1 \leq_0 T_2$ provided that, for every environment E for both T_1 and T_2, $tdists(T_1 \| E) \subseteq tdists(T2 \| E)$.*

We obtain the following theorems.

Theorem 1. *Suppose that T_1 and T_2 are comparable task-PIOAs such that $T_1 \leq_0 T_2$, and T_3 is a task-PIOA that is compatible with each of T_1 and T_2. Then $T_1 \parallel T_3 \leq_0 T_2 \parallel T_3$.*

Theorem 2. *Suppose that T_1 and T_2 are comparable task-PIOAs such that $T_1 \leq_0 T_2$. Suppose that S is a set of output actions of both T_1 and T_2. Then $hide(T_1, S) \leq_0 hide(T_2, S)$.*

Definition 12. *Let $T_1 = (P_1, R_1)$ and $T_2 = (P_2, R_2)$ be two comparable closed task-PIOAs. Let R be a relation from $Disc(Execs^*(P_1))$ to $Disc(Execs^*(P_2))$ such that if $\epsilon_1 R \epsilon_2$, then $tdist(\epsilon_1) = tdist(\epsilon_2)$. Then R is a simulation from T_1 to T_2 if there exists $c : (R_1^* \times R_1) \to R_2^*$ such that the start and step conditions given below hold.*
1. Start condition: $\delta(\bar{q}_1) R \delta(\bar{q}_2)$.
2. Step condition: If $\epsilon_1 R \epsilon_2$, $\rho \in R_1^*$, ϵ_1 is consistent with ρ, ϵ_2' is consistent with $full(c)(\rho)$, and $T \in R_1$, then $\epsilon_1' E(R) \epsilon_2'$, where $\epsilon_1' = apply(\epsilon_1, T)$ and $\epsilon_2' = apply(\epsilon_2, c(\rho, T))$.

Theorem 3. *Let T_1 and T_2 be two comparable closed task-PIOAs. If there is a simulation relation R from T_1 to T_2, then $tdists(T_1) \subseteq tdists(T_2)$.*

Corollary 1. *Let T_1 and T_2 be two comparable task-PIOAs. Suppose that, for every environment E for both T_1 and T_2, there is a simulation relation R from $T_1 \| E$ to $T_2 \| E$. Then $T_1 \leq_0 T_2$.*

Corollary 2. *Let $T_1 = (P_1, R_1)$ and $T_2 = (P_2, R_2)$ be two comparable closed task- PIOAs. Let R be a relation from $Disc(Execs^*(P_1))$ to $Disc(Execs^*(P_2))$, satisfying the following condition: if $\epsilon_1 R \epsilon_2$ then $tdist(\epsilon_1) = tdist(\epsilon_2)$. Let $c : (R_1^* \times R_1) \rightarrow R_2^*$. Suppose further that the following conditions hold. 1. Start condition: $\delta(\bar{q}_1) R \delta(\bar{q}_1)$.*
2. Step condition: If $\epsilon_1 R \epsilon_2$, $\rho_1 \in R_1^*$, ϵ_1 is consistent with ρ_1, ϵ_2' is consistent with $full(c)(\rho_1)$, and $T \in R_1$, then there exist the following.

- *Probability measure p on countable index set I,*
- *Probability measures $\epsilon_{1,j}'$, $j \in I$ on finite executions of P_1, and*
- *Probability measures $\epsilon_{2,j}'$, $j \in I$ on finite executions of P_2,*

such that:

- *For each $j \in I$, $\epsilon_{1,j}' R \epsilon_{2,j}'$,*
- *$Sigma_{j \in I} p(j)(\epsilon_{1,j}') = apply(\epsilon_1, T)$, and*
- *$Sigma_{j \in I} p(j)(\epsilon_{2,j}') = apply(\epsilon_2, c(\rho_1, T))$.*

Then R is a simulation relation from T_1 to T_2 using c.

4.6 Security Definitions

We define the security notion on task-PIOA considering the synchronous and asynchronous schedule as described hereafter.

Definition 13. [Perfect Implementation:] *Let* Env, Real *and* Ideal *be an environment task-PIOA, a real protocol task-PIOA system and an ideal functionality task-PIOA system, respectively. Let* $\mathtt{sch}$ *be some (synchronous or asynchronous) schedule. We say that* Real *perfectly implements* Ideal *under some (synchronous or asynchronous) schedule (or* Real $\leq_0^{\mathtt{sch.}}$ Ideal*), if* $trace(\mathrm{Real}||\mathrm{Env}) = trace(\mathrm{Ideal}||\mathrm{Env})$ *for every environment* Env *under a synchronous or asynchronous schedule.*

Definition 14. [Perfect Hybrid Implementation:] *Let* Hybrid *be a real protocol task-PIOA system with the hybrid model. We say that* Hybrid *perfectly hybrid-implements* Ideal *under some (synchronous or asynchronous) schedule (or* Hyb. $\leq_0^{\mathtt{sch.}}$ Ideal}*), if* $trace(\mathrm{Hybrid}||\mathrm{Env}) = trace(\mathrm{Ideal}||\mathrm{Env})$ *for every environment* Env *under some (synchronous or asynchronous) schedule.*

Chapter 5

Semantic Securities for KEM and DEM

This chapter defines the three NMs for KEM and proves equivalence among the three NMs. In addition, we prove equivalence between IND and NM. We then provide the IND and NM of DEM and prove equivalence between IND and NM.

5.1 Three NMs for KEM

The NM of KEM is defined as three NMs: a simulation based NM, comparison based NM, and parallel chosen-ciphertext attack based NM.

5.1.1 Simulation Based NM

KEM Σ is called "SNM-ATK-KEM" in the sense that Σ is secure in *simulation based NM* (SNM) for each attack type ATK $\in$ {CPA, CCA1, CCA2}.

Definition 15. *Let* Σ *be* KEM*,* Rel *be a relation,* $A = (A_1, A_2)$ *be an adversary,* $S = (S_1, S_2)$ *be an algorithm (the "simulator"), and* $k \in \mathtt{N}$ *be the security parameter. For* ATK $\in$ {CPA, CCA1, CCA2}*, we define* $\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{A,S,\Sigma}(Rel, k)$ *in Fig. 5.1. We say that* Σ *is SNM-ATK-KEM, if for any* *adversary $A \in \mathcal{P}$ and all relations Rel computable in $\mathcal{P}$, there exists simulator $S \in \mathcal{P}$ such that $\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{A,S,\Sigma}(Rel,k)$ is negligible in k, where* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$ *and $\mathcal{P}$ denotes a class of polynomial-time bounded machines.*

Note that adversary A_2 is not allowed to pose the challenge ciphertext C^* to its decryption oracle in the case of CCA2 and we require that $C^* \notin \boldsymbol{C}$.

In the previous NM definitions [41, 42], the adversary can select the key space. As mentioned in the Introduction, it is a serious problem that the definitions are available only for a few types of KEM schemes. Therefore, the revised point in this paper is to free the key space of the old version definition in $\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{A,\Sigma}(Rel,k)$.

In the attack scenario of SNM for PKE (SNM-PKE), the adversary can decide the message space [10]. Note that such a message space in the scenario is introduced to make SNM-PKE compatible with IND-PKE, i.e., to make SNM-PKE imply IND-PKE, in the attack scenario in which the adversary can decide a pair of messages (a message space).

In contrast, in the IND-KEM attack scenario, a correct key or a random value along with the target ciphertext is given to the adversary. To make SNM-KEM compatible with IND-KEM, i.e., to make SNM-KEM imply IND-KEM, the SNM-KEM attack scenario herein gives the adversary a randomly-ordered pair comprising a correct key and a random value.

Here, if KEM Σ is not IND(-ATK), i.e., adversary A can distinguish (C^*, K^*) and (C^*, R^*)), Σ is not NM(-ATK). For example, A guesses K^* from X, sets $Rel(K^*, K')$ iff $lsb(K^*) = lsb(K')$, and randomly searches for C' such that $(K', C') \overset{\mathsf{R}}{\leftarrow} \mathcal{E}(pk)$ and $lsb(K^*) = lsb(K')$).

Two additional minor differences between SNM-KEM and SNM-PKE are given hereafter.

1. Simulator S also obtains access to the decryption oracle when ATK allows it to do so.

2. Relation R utilizes state information s calculated not by A_1 or S_1 but by A_2 or S_2 in SNM-KEM.

The difference between the NM-KEM herein and wNM-KEM proposed by Herrantz et al. [35] is whether or not adversary A_2 can gain key information X (this includes the order of key K^* and a random string R (or another random string R^*)). Information X in the definition of the SNM-KEM (PNM-KEM, and CNM-KEM) described herein plays a similar role to the message space in the NM definitions in [10, 3] for PKE.

5.1.2 Comparison Based NM

KEM Σ is called "CNM-ATK-KEM" in the sense that Σ is secure in *comparison based NM* (CNM) for each attack type $\text{ATK} \in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$.

Definition 16. *Let Σ be* KEM, *$A = (A_1, A_2)$ be an adversary, and $k \in \mathbb{N}$ be the security parameter. For* $\text{ATK} \in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$, *we define* $\mathtt{Adv}_{A,\Sigma}^{\mathsf{CNM\text{-}ATK}}(k)$ *in Fig. 5.2. We say that Σ is CNM-ATK-KEM, if for any adversary $A \in \mathcal{P}$,* $\mathtt{Adv}_{A,\Sigma}^{\mathsf{CNM\text{-}ATK}}(k)$ *is negligible in k, where* $\text{ATK} \in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$ *and $\mathcal{P}$ denotes a class of polynomial-time bounded machines.*

Note that adversary A_2 is not allowed to ask its oracle to decrypt challenge ciphertext C^* for CCA2 and we require that $C^* \notin \boldsymbol{C}$.

The revised point is to free the key space of the old version definitions in $\mathtt{Expt}_{A,\Sigma}^{\mathsf{CNM\text{-}ATK}}(k)$ and $\widetilde{\mathtt{Expt}}_{A,\Sigma}^{\mathsf{CNM\text{-}ATK}}(k)$.

Similar to SNM-KEM, the CNM-KEM's attack scenario herein gives the adversary a randomly-ordered pair comprising a correct key and a random value to make CNM-KEM compatible with IND-KEM.

5.1.3 Parallel Chosen-Ciphertext Attack Based NM

KEM Σ is called "PNM-ATK-KEM" in the sense that Σ is secure in *parallel chosen-ciphertext attack based NM* (PNM) for each attack type $\text{ATK} \in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$.

Definition 17. *Let Σ be a* KEM, *$A = (A_1, A_2, A_3)$ be an adversary, and $k \in \mathbb{N}$ be the security parameter. For* $\text{ATK} \in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$, *we de-*

fine $\mathtt{Adv}^{\mathsf{PNM\text{-}ATK}}_{A,\Sigma}(k)$ *in Fig. 5.3. We say that* Σ *is PNM-ATK-KEM, if for any adversary* $A \in \mathcal{P}$, $\mathtt{Adv}^{\mathsf{PNM\text{-}ATK}}_{A,\Sigma}(k)$ *is negligible in k, where k is the security parameter,* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$*, and* $\mathcal{P}$ *denotes a class of polynomial-time bounded machines.*

Note that adversary A_2 is not allowed to ask its oracle to decrypt challenge ciphertext C^* for CCA2 and we require that $C^* \notin \boldsymbol{C}$.

The revised point is to free the key space of the old version definitions in $\mathtt{Expt}^{\mathsf{PNM\text{-}ATK}}_{A,\Sigma}(k)$.

In the PNM definition, the NM property is captured by IND under the parallel chosen-ciphertext attack such that A_2 outputs a vector of ciphertext $\boldsymbol{C}$ and its decryption result $\boldsymbol{K}$ is given to A_3.

5.2 Equivalence among Three NM Definitions

Here, we prove the equivalence of the three NM definitions.

Theorem 1. *For any* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$*, if* KEM Σ *is CNM-ATK-KEM, then* Σ *is SNM-ATK-KEM.*

Theorem 2. *For any* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$*, if* KEM Σ *is SNM-ATK-KEM, then* Σ *is PNM-ATK-KEM.*

Theorem 3. *For any* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$*, if* KEM Σ *is PNM-ATK-KEM, then* Σ *is CNM-ATK-KEM.*

5.2.1 Proof of Theorem 1

Proof. We prove that KEM Σ is not CNM-ATK-KEM if Σ is not SNM-ATK-KEM. More precisely, we show that if adversary *Adv* and relation *Rel* exist such that $\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{Adv,Sim,\Sigma}(Rel, k)$ is not negligible in k for any simulator Sim, then there exists adversary B such that $\mathtt{Adv}^{\mathsf{CNM\text{-}ATK}}_{B,\Sigma}(k)$ is not negligible in k, where k is the security parameter and $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$.

Let $A = (A_1, A_2)$ be an adversary to SNM-ATK. First, we construct a CNM-ATK adversary, $B = (B_1, B_2)$, using SNM-ATK adversary *Adv* in Fig.

5.4. From the construction of B, we obtain the following equivalence for all $k \in \mathtt{N}$:

$$\Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{Adv,\Sigma}(Rel,k) = 1] = \Pr[\mathtt{Expt}^{\mathsf{CNM\text{-}ATK}}_{B,\Sigma}(k) = 1]. \qquad (5.1)$$

We then construct SNM-ATK simulator $\hat{Sim} = (\hat{Sim}_1, \hat{Sim}_2)$ using SNM-ATK adversary Adv as shown in Fig. 5.5.

From the construction of B using Adv, and the construction of $\hat{Sim}$, we obtain the following equivalence for all $k \in \mathtt{N}$:

$$\Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{\hat{Sim},\Sigma}(Rel,k) = 1] = \Pr[\widetilde{\mathtt{Expt}}^{\mathsf{CNM\text{-}ATK}}_{B,\Sigma}(k) = 1]. \qquad (5.2)$$

Here, note that, even if $A_2^{O_2}$ outputs $\boldsymbol{C}$ with $C^* \in \boldsymbol{C}$, $B_2^{O_2}$ outputs ciphertext vector $\boldsymbol{C}$ and $\widetilde{\mathtt{Expt}}^{\mathsf{CNM\text{-}ATK}}_{B,\Sigma}(k)$ returns 0 because $C^* \in \boldsymbol{C}$. $\hat{Sim}_2{}^{O_2}$ returns $\perp$ and $\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{\hat{Sim},\Sigma}(Rel,k)$ returns 0 (A problem regarding this note was investigated in [39]).

The assumption (for contradiction) is that, for any Sim, the advantage $\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{Adv,Sim,\Sigma}(Rel,k) > \mu(k)$ implies $\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{Adv,\hat{Sim},\Sigma}(Rel,k) > \mu(k)$ (for a specific $\hat{Sim}$). From this inequality and Eqs.(5.1) and (5.2), we obtain

$$\begin{aligned}
&\mathtt{Adv}^{\mathsf{CNM\text{-}ATK}}_{B,\Sigma}(k)\\
&= \Pr[\mathtt{Expt}^{\mathsf{CNM\text{-}ATK}}_{B,\Sigma}(k) = 1] - \Pr[\widetilde{\mathtt{Expt}}^{\mathsf{CNM\text{-}ATK}}_{B,\Sigma}(k) = 1]\\
&= \Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{Adv,\Sigma}(Rel,k) = 1] - \Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{\hat{Sim},\Sigma}(Rel,k) = 1]\\
&= \mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{Adv,\hat{Sim},\Sigma}(Rel,k) > \mu(k).
\end{aligned}$$

□

5.2.2 Proof of Theorem 2

Proof. We prove that KEM Σ is not SNM-ATK-KEM if Σ is not PNM-ATK-KEM. More precisely, we show that if there exists adversary Adv such that $\mathtt{Adv}^{\mathsf{PNM\text{-}ATK}}_{Adv,\Sigma}(k)$ is not negligible in k, then adversary B and relation Rel exist for any simulator Sim such that $\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{B,Sim,\Sigma}(Rel,k)$ is not negligible in k where k is a security parameter and $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$.

Let $A = (A_1, A_2, A_3)$ be an adversary for PNM-ATK. First, we construct SNM-ATK adversary $B = (B_1, B_2)$ and relation Rel using PNM-ATK adversary Adv as shown in Fig. 5.6. Here, we say event Bad occurs iff Y is not an element of X. From the construction of B, we obtain the following equivalence for all $k \in \mathtt{N}$:

$$\Pr[\mathtt{Expt}^{\mathsf{PNM\text{-}ATK}}_{Adv,\Sigma}(k) = 1] = \Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{B,\Sigma}(Rel,k) = 1] \tag{5.3}$$

Using Eq.(5.4), we show that given relation Rel, for any simulator Sim, the success probability of $\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{Sim,\Sigma}(Rel,k)$ is at most $\frac{1}{2}$.

$$\begin{aligned}
&\Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{Sim,\Sigma}(Rel,k) = 1] \\
&= \Pr[g = b \wedge \neg\mathtt{Bad}] \\
&= \Pr[b = 0 \wedge g = 0 \wedge \neg\mathtt{Bad}] + \Pr[b = 1 \wedge g = 1 \wedge \neg\mathtt{Bad}] \\
&= \Pr[b = 0 \wedge \neg\mathtt{Bad}] \times \Pr[g = 0 | b = 0 \wedge \neg\mathtt{Bad}] \\
&\quad +\Pr[b = 1 \wedge \neg\mathtt{Bad}] \times \Pr[g = 1 | b = 1 \wedge \neg\mathtt{Bad}] \\
&\le \frac{1}{2} \times \Pr[g = 0 | b = 0 \wedge \neg\mathtt{Bad}] + \frac{1}{2} \times \Pr[g = 1 | b = 1 \wedge \neg\mathtt{Bad}] \\
&\qquad \text{(here } b \text{ and } \mathtt{Bad} \text{ are independent of } g\text{)} \\
&= \frac{1}{2} \times (\Pr[g = 0] + \Pr[g = 1]) = \frac{1}{2}
\end{aligned} \tag{5.4}$$

By applying Eqs. (5.3), (5.4) and the above-mentioned assumption that $\mathtt{Adv}^{\mathsf{PNM\text{-}ATK}}_{Adv,\Sigma}(k) > \mu(k)$, we obtain:

$$\begin{aligned}
&\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{B,Sim,\Sigma}(Rel,k) \\
&= \Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{B,\Sigma}(Rel,k) = 1] - \Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{Sim,\Sigma}(Rel,k) = 1] \\
&\ge \Pr[\mathtt{Expt}^{\mathsf{PNM\text{-}ATK}}_{Adv,\Sigma}(k) = 1] - \frac{1}{2} \\
&= \mathtt{Adv}^{\mathsf{PNM\text{-}ATK}}_{Adv,\Sigma}(k) \ > \mu(k).
\end{aligned}$$

□

5.2.3 Proof of Theorem 3

Proof. We prove that KEM Σ is not PNM-ATK-KEM if Σ is not CNM-ATK-KEM. More precisely, we show that if there exists adversary *Adv* such that $\mathtt{Adv}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k)$ is not negligible in k, then there exists adversary B such that $\mathtt{Adv}^{\textsf{PNM-ATK}}_{B,\Sigma}(k)$ is not negligible in k where k is the security parameter and ATK $\in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$.

Let $A = (A_1, A_2)$ be an adversary for CNM-ATK. We construct PNM-ATK adversary $B = (B_1, B_2, B_3)$ using CNM-ATK adversary *Adv* as shown in Fig. 5.7. From the construction of B, we obtain

$$\begin{aligned}
&\Pr[\mathtt{Expt}^{\textsf{PNM-ATK}}_{B,\Sigma}(k) = 1] \\
&\quad = \Pr[b = g] \\
&\quad = \Pr[b = 0 \wedge g = 0] + \Pr[b = 1 \wedge g = 1] \\
&\quad = \Pr[b = 0] \times \Pr[g = 0 | b = 0] + \Pr[b = 1] \times \Pr[g = 1 | b = 1] \\
&\quad = \frac{1}{2}\Pr[\mathtt{Expt}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) = 1] + \frac{1}{2}(1 - \Pr[\widetilde{\mathtt{Expt}}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) = 1]) \\
&\quad = \frac{1}{2}(\Pr[\mathtt{Expt}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) = 1] - \Pr[\widetilde{\mathtt{Expt}}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) = 1]) + \frac{1}{2}.
\end{aligned}$$

That is,
$$\begin{aligned}
&\Pr[\mathtt{Expt}^{\textsf{PNM-ATK}}_{B,\Sigma}(k) = 1] - \frac{1}{2} \\
&\quad = \frac{1}{2}(\Pr[\mathtt{Expt}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) = 1] - \Pr[\widetilde{\mathtt{Expt}}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) = 1]) \\
&\quad = \frac{1}{2}\mathtt{Adv}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k). \qquad (5.5)
\end{aligned}$$

By applying Eq.(5.5) and the above-mentioned assumption that $\mathtt{Adv}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) > \mu(k)$, we obtain

$$\mathtt{Adv}^{\textsf{PNM-ATK}}_{B,\Sigma}(k) = \frac{1}{2}\mathtt{Adv}^{\textsf{CNM-ATK}}_{Adv,\Sigma}(k) > \mu(k)/2.$$

□

5.2.4 Equivalence of the Three NM Definitions

From Theorems 1, 2 and 3, we immediately obtain the equivalence of the three NM definitions, SNM-ATK-KEM, CNM-ATK-KEM, and PNM-ATK-KEM. Hereafter, we use NM-ATK-KEM to refer to the three NM definitions. If it is clear that NM-ATK-KEM is used for KEM, we refer to it simply as NM-ATK.

5.3 IND-CCA2 KEM is Equivalent to NM-CCA2 KEM

This section shows that NM is equivalent to IND for KEM against adaptive chosen ciphertext attacks (CCA2). For PKE, it has already been proven that NM is equivalent to IND against CCA2 [3].

Theorem 4. KEM Σ *is NM-CCA2-KEM, if and only if* Σ *is IND-CCA2-KEM.*

Proof. To prove this theorem, it is sufficient to show that PNM-CCA2-KEM is equivalent to IND-CCA2-KEM. It is trivial from the definition that KEM Σ is not IND-CCA2-KEM if Σ is not PNM-CCA2-KEM. The opposite direction, i.e., Σ is not PNM-CCA2-KEM if Σ is not IND-CCA2-KEM, is also easy to prove as indicated hereafter. Let $A = (A_1, A_2)$ be an attacker for IND-CCA2-KEM. We then construct attacker $B = (B_1, B_2, B_3)$ for PNM-CCA2-KEM using *Adv* such that B_1 executes A_1, and B_2 executes A_2 which outputs g and outputs $(s_2, \boldsymbol{C})$ such that $s_2 \leftarrow g$ and $C^* \notin \boldsymbol{C}$. B_3 outputs $s_2(= g)$ regardless of the value of $\boldsymbol{K}$. Clearly, B is an attacker of PNM-CCA2-KEM with the same advantage as *Adv* for IND-CCA2-KEM. □

5.4 IND and NM for DEM

This section explains the definition of IND and NM for DEM.

5.4.1 IND for DEM

The advantage of indistinguishability of DEM (we use "IND-DEM") following [37] is stated in Fig.5.8. In this thesis, we also use IND-PX-CY-DEM to describe the security notion of the IND of DEM against {X, Y} ∈ {0, 1, 2}.

Definition 18. *Let* Σ' *be a* DEM *over message space* M*,* $A = (A_1, A_2)$ *be an adversary, and* $k \in \mathtt{N}$ *be the security parameter. For* $\{X, Y\} \in \{0, 1, 2\}$*,* $\mathtt{Adv}^{\mathsf{IND\text{-}PX\text{-}CY}}_{A,\Sigma'}(k)$ *is defined in Fig. 5.8. We say that* Σ' *is IND-PX-CY-DEM, if for any adversary* $A \in \mathcal{P}$*,* $\mathtt{Adv}^{\mathsf{IND\text{-}PX\text{-}CY}}_{A,\Sigma'}(k)$ *is negligible in k where* $\{X, Y\} \in \{0, 1, 2\}$*, and* $\mathcal{P}$ *denotes a class of polynomial-time bounded machines.*

Note that, the length of x_0 equals the length of x_1, i.e., $|x_0| = |x_1|$. Furthermore, when $Y = 2$, we insist that A_2 does not ask for the decryption of challenge ciphertext y.

5.4.2 NM for DEM

We state the formal definition of NM for DEM in Fig.5.9 following Bellare[10] and Katz[37], which we call NM-DEM. We also use NM-PX-CY-DEM to describe NM for DEM against several types of attacks where {X, Y} ∈ {0, 1, 2}.

Definition 19. *Let* $\Sigma' = (\mathcal{E}', \mathcal{D}')$ *be a* DEM *over message space* M*,* $A = (A_1, A_2)$ *be an adversary, and* $k \in \mathtt{N}$ *be the security parameter. For* $\{X, Y\} \in \{0, 1, 2\}$*,* $\mathtt{Adv}^{\mathsf{NM\text{-}PX\text{-}CY}}_{A,\Sigma'}(k)$ *is defined in Fig. 5.9. We say that* Σ' *is NM-PX-CY-DEM, if for any adversary* $A \in \mathcal{P}$*,* $\mathtt{Adv}^{\mathsf{NM\text{-}PX\text{-}CY}}_{A,\Sigma'}(k)$ *is negligible in k, where* $\{X, Y\} \in \{0, 1, 2\}$*, and* $\mathcal{P}$ *denotes a class of polynomial-time bounded machines.*

Informally, we describe the secure notion of NM for DEM. In Fig.5.9, O_i (or $O_i{}'$) = $\perp$ where $i \in \{1,2\}$. This means that O_i (or $O_i{}'$), which takes on any input, returns the empty string $\perp$, and R is some relation. We require that $|x| = |\widetilde{x}|$ for all x and $\widetilde{x}$ in the support of M. We also require that the vector of ciphertexts, $\boldsymbol{y}$, output by A_2 should be non-empty and $y \notin \boldsymbol{y}$. Furthermore, when $Y = 2$, we insist that A_2 does not ask for the decryption of y.

At the first stage of the attack, adversary A_1 outputs distribution M over messages along with state information s. Two messages, x and $\widetilde{x}$ are chosen at random according to M, and x is encrypted to give ciphertext y. In the second stage of the attack, y and s are given to adversary A_2. A_2 outputs relation R and a vector of ciphertexts $\boldsymbol{y}$ such that $y \notin \boldsymbol{y}$ (we require that $\widetilde{y} \notin \widetilde{\boldsymbol{y}}$ in $\widetilde{\mathtt{Expt}}^{\mathsf{NM\text{-}PX\text{-}CY}}_{A,\Sigma'}(k)$).

Σ' is NM secure in the sense of NM-PX-CY for {X, Y} ∈ {0, 1, 2} if $\mathtt{Adv}^{\mathsf{NM\text{-}PX\text{-}CY}}_{A,\Pi_{\Sigma'}}(k)$ is negligible for any PPT adversary A. That is, we say Σ' is NM secure if for every PPT algorithm A, the probability that $R(\widetilde{x}, \boldsymbol{x})$ is true is at most negligibly different from the probability that $R(x, \boldsymbol{x})$ is true.

5.5 IND-P2-C2 DEM is Equivalent to NM-P2-C2 DEM

The two above security notions of DEM yield the following Theorem 5.

Theorem 5. *Encryption scheme Σ' is secure in the sense of NM-P2-C2 if and only if Σ' is secure in the sense of IND-P2-C2.*

Proof. We prove the equivalence between IND and NM for DEM hereafter.

(”only if” part) Let A be an adversary attacking Σ' in the sense of IND-P2-C2-DEM. We construct adversary $B = (B_1, B_2)$ attacking Σ' in the sense of NM-P2-C2-DEM in Fig.5.10.

It is not difficult to see that $\Pr[\widetilde{\mathtt{Expt}}^{\mathsf{NM\text{-}P2\text{-}C2}}_{B,\Sigma'}(k)] = \frac{1}{2}$ so that

$$\mathtt{Adv}^{\mathsf{NM\text{-}P2\text{-}C2}}_{B,\Sigma'}(k) = \mathtt{Adv}^{\mathsf{IND\text{-}P2\text{-}C2}}_{A,\Sigma'}(k).$$

Since Σ' is secure in the sense of NM-P2-C2-DEM, $\mathtt{Adv}^{\mathsf{IND\text{-}P2\text{-}C2}}_{A,\Sigma'}(k)$ is negligible and the theorem follows.

(”if” part) This direction is the exact counterpart of [37], and we repeat essentially the same proof here for completeness. Let A be an adversary attacking Σ' in the sense of NM-P2-C2-DEM. We define adversary B attacking Σ' in the sense of IND-P2-C2-DEM in Fig.5.11.

We note that the probability that B returns 0 given that y is an encryption of x_0 is exactly $\Pr[\mathtt{Expt}_{A,\Sigma'}^{\textsf{NM-P2-C2}}(k)]$ while the probability that B returns 0 given that y is an encryption of x_1 is exactly $\Pr[\widetilde{\mathtt{Expt}}_{A,\Sigma'}^{\textsf{NM-P2-C2}}(k)]$. Thus,

$$\mathtt{Adv}_{B,\Sigma'}^{\textsf{IND-P2-C2}}(k) = \mathtt{Adv}_{A,\Sigma'}^{\textsf{NM-P2-C2}}(k)$$

and hence $\mathtt{Adv}_{A,\Sigma'}^{\textsf{NM-P2-C2}}(k)$ is negligible. $\square$

$$\mathtt{Adv}^{\mathsf{SNM\text{-}ATK}}_{A,S,\Sigma}(Rel,k) \equiv \Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{A,\Sigma}(Rel,k)=1] - \Pr[\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{S,\Sigma}(Rel,k)=1]$$

where $\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{A,\Sigma}(Rel,k)$:

$$(pk,sk) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^k); s_1 \overset{\mathsf{R}}{\leftarrow} A_1^{O_1}(pk)$$
$$(K^*,C^*) \overset{\mathsf{R}}{\leftarrow} \mathcal{E}(pk); R \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)}$$
$$b \overset{\mathsf{U}}{\leftarrow} \{0,1\}$$
$$X \leftarrow (r_0,r_1) \text{where} \begin{cases} r_0 \leftarrow K^* \text{ and } r_1 \leftarrow R, \text{ if } b=0 \\ r_0 \leftarrow R \text{ and } r_1 \leftarrow K^*, \text{ if } b=1 \end{cases}$$
$$(s_2,\boldsymbol{C}) \overset{\mathsf{R}}{\leftarrow} A_2^{O_2}(X,s_1,C^*)$$
$$\boldsymbol{K} \leftarrow \mathcal{D}(sk,\boldsymbol{C})$$
$$\text{return 1, iff } Rel(K^*,\boldsymbol{K},s_2)$$

$\mathtt{Expt}^{\mathsf{SNM\text{-}ATK}}_{S,\Sigma}(Rel,k)$:

$$(pk,sk) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^k); s_1 \overset{\mathsf{R}}{\leftarrow} S_1^{O_1}(pk)$$
$$R^* \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)}; R \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)}$$
$$b \overset{\mathsf{U}}{\leftarrow} \{0,1\}$$
$$X \leftarrow (r_0,r_1) \text{where} \begin{cases} r_0 \leftarrow R^* \text{ and } r_1 \leftarrow R, \text{ if } b=0 \\ r_0 \leftarrow R \text{ and } r_1 \leftarrow R^*, \text{ if } b=1 \end{cases}$$
$$(s_2,\boldsymbol{C}) \overset{\mathsf{R}}{\leftarrow} S_2^{O_2}(X,s_1); \boldsymbol{K} \leftarrow \mathcal{D}(sk,\boldsymbol{C})$$
$$\text{return 1, iff } Rel(R^*,\boldsymbol{K},s_2)$$

and
If ATK = CPA, then $O_1 = \perp$ and $O_2 = \perp$.
If ATK = CCA1, then $O_1 = \mathcal{D}(sk,\cdot)$ and $O_2 = \perp$.
If ATK = CCA2, then $O_1 = \mathcal{D}(sk,\cdot)$ and $O_2 = \mathcal{D}(sk,\cdot)$.

Figure 5.1: Advantage of SNM-ATK-KEM

$$\mathtt{Adv}^{\mathsf{CNM\text{-}ATK}}_{A,\Sigma}(k) \equiv \Pr[\mathtt{Expt}^{\mathsf{CNM\text{-}ATK}}_{A,\Sigma}(k)=1] - \Pr[\widetilde{\mathtt{Expt}}^{\mathsf{CNM\text{-}ATK}}_{A,\Sigma}(k)=1]$$

where $\mathtt{Expt}^{\mathsf{CNM\text{-}ATK}}_{A,\Sigma}(k)$:

$$(pk,sk) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^k); s \overset{\mathsf{R}}{\leftarrow} A_1^{O_1}(pk)$$
$$(K^*,C^*) \overset{\mathsf{R}}{\leftarrow} \mathcal{E}(pk); R \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)}$$
$$b \overset{\mathsf{U}}{\leftarrow} \{0,1\}$$
$$X \leftarrow (r_0,r_1), \text{where} \begin{cases} r_0 \leftarrow K^* \text{ and } r_1 \leftarrow R, & \text{if } b=0 \\ r_0 \leftarrow R \text{ and } r_1 \leftarrow K^*, & \text{if } b=1 \end{cases}$$
$$(Rel,\boldsymbol{C}) \overset{\mathsf{R}}{\leftarrow} A_2^{O_2}(X,s,C^*); \boldsymbol{K} \leftarrow \mathcal{D}(sk,\boldsymbol{C})$$
$$\text{return 1, iff } Rel(K^*,\boldsymbol{K})$$

$\widetilde{\mathtt{Expt}}^{\mathsf{CNM\text{-}ATK}}_{A,\Sigma}(k)$:

$$(pk,sk) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^k); s \overset{\mathsf{R}}{\leftarrow} A_1^{O_1}(pk)$$
$$(K^*,C^*) \overset{\mathsf{R}}{\leftarrow} \mathcal{E}(pk); R^* \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)}$$
$$R \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)}; b \overset{\mathsf{U}}{\leftarrow} \{0,1\}$$
$$X \leftarrow (r_0,r_1), \text{where} \begin{cases} r_0 \leftarrow R^* \text{ and } r_1 \leftarrow R, & \text{if } b=0 \\ r_0 \leftarrow R \text{ and } r_1 \leftarrow R^*, & \text{if } b=1 \end{cases}$$
$$(Rel,\boldsymbol{C}) \overset{\mathsf{R}}{\leftarrow} A_2^{O_2}(X,s,C^*); \boldsymbol{K} \leftarrow \mathcal{D}(sk,\boldsymbol{C})$$
$$\text{return 1, iff } Rel(R^*,\boldsymbol{K})$$

and
If ATK = CPA, then $O_1 = \perp$ and $O_2 = \perp$.
If ATK = CCA1, then $O_1 = \mathcal{D}(sk,\cdot)$ and $O_2 = \perp$.
If ATK = CCA2, then $O_1 = \mathcal{D}(sk,\cdot)$ and $O_2 = \mathcal{D}(sk,\cdot)$.

Figure 5.2: Advantage of CNM-ATK-KEM

$$\mathtt{Adv}^{\mathsf{PNM\text{-}ATK}}_{A,\Sigma}(k) \equiv \Pr[\mathtt{Expt}^{\mathsf{PNM\text{-}ATK}}_{A,\Sigma}(k) = 1] - \frac{1}{2}$$

where $\mathtt{Expt}^{\mathsf{PNM\text{-}ATK}}_{A,\Sigma}(k)$:

```
$(pk, sk) \overset{R}{\leftarrow} \mathcal{G}(1^k); s_1 \overset{R}{\leftarrow} A_1^{O_1}(pk)$
$(K^*, C^*) \overset{R}{\leftarrow} \mathcal{E}(pk); R \overset{U}{\leftarrow} \{0,1\}^{l(k)}$
$b \overset{U}{\leftarrow} \{0,1\}$
$X \leftarrow \begin{cases} K^*, \text{ if } b = 0 \\ R, \text{ if } b = 1 \end{cases}$
$(s_2, \boldsymbol{C}) \overset{R}{\leftarrow} A_2^{O_2}(X, s_1, C^*)$
$\boldsymbol{K} \leftarrow \mathcal{D}(sk, \boldsymbol{C}); g \overset{R}{\leftarrow} A_3(s_2, \boldsymbol{K})$
return 1, iff $g = b$
```

and
If ATK = CPA, then $O_1 = \perp$ and $O_2 = \perp$.
If ATK = CCA1, then $O_1 = \mathcal{D}(sk, \cdot)$ and $O_2 = \perp$.
If ATK = CCA2, then $O_1 = \mathcal{D}(sk, \cdot)$ and $O_2 = \mathcal{D}(sk, \cdot)$.

Figure 5.3: Advantage of PNM-ATK-KEM

$B_1^{O_1}(pk)$	$B_2^{O_2}(X, s, C^*)$
$t_1 \overset{R}{\leftarrow} A_1^{O_1}(pk)$	$(s_2, \boldsymbol{C}) \overset{R}{\leftarrow} A_2^{O_2}(X, s, C^*)$
$s \leftarrow t_1$	Define Rel' by $Rel'(a, \boldsymbol{b}) = 1$,
return s	iff $Rel(a, \boldsymbol{b}, s_2) = 1$
	return $(Rel', \boldsymbol{C})$

Figure 5.4: CNM-ATK adversary B using SNM-ATK adversary Adv.

```
$\underline{S\hat{i}m_1^{O_1}(pk)}$
$t_1 \overset{R}{\leftarrow} A_1^{O_1}(pk)$
$s_1 \leftarrow t_1$
return $s_1$

$\underline{S\hat{i}m_2^{O_2}(X, s_1)}$
$(K^*, C^*) \overset{R}{\leftarrow} \mathcal{E}(pk)$
$(s_2, \boldsymbol{C}) \overset{R}{\leftarrow} A_2^{O_2}(X, s_1, C^*)$
If $C^* \in \boldsymbol{C}$, then return $\perp$.
Otherwise, return $(s_2, \boldsymbol{C})$.
```

Figure 5.5: SNM-ATK simulator $S\hat{i}m$ using SNM-ATK adversary *Adv*.

```
$\underline{B_1^{O_1}(pk)}$
        $t_1 \overset{R}{\leftarrow} A_1^{O_1}(pk); s_1 \leftarrow t_1;$ return $s_1$

$\underline{B_2^{O_2}(X, s_1, C^*)\text{, where } s_1 = t_1 \text{ and } X = (r_0, r_1)}$
        $(t_2, \boldsymbol{C}) \overset{R}{\leftarrow} A_2^{O_2}(r_0, t_1, C^*)$
        Choose random coins $\sigma$ for $A_3$.
        $s_2 \leftarrow (t_2, \sigma, X);$ return $(s_2, \boldsymbol{C})$

$\underline{Rel(Y, \boldsymbol{K}, s_2)\text{, where } s_2 = (t_2, \sigma, X)}$
        If $Y$ is not an element of $X$, return 0.
        If $Y = r_0$, then $b = 0$. Otherwise, $b = 1$.
        $g \leftarrow A_3(t_2, \boldsymbol{K}; \sigma);$ return 1, iff $b = g$
```

Figure 5.6: SNM-ATK adversary *B* and Relation *Rel* using PNM-ATK adversary *Adv*.

```
$\underline{B_1^{O_1}(pk)}$

        $t \overset{R}{\leftarrow} A_1^{O_1}(pk); s_1 \leftarrow t;$ return $s_1$

$\underline{B_2^{O_2}(X, s_1, C^*), \text{ where } s_1 = t \text{ and } X = K^* \text{ or } R}$

        $R' \overset{U}{\leftarrow} \{0,1\}^{l(k)}; c \overset{U}{\leftarrow} \{0,1\}$
        $X' \leftarrow \begin{cases} (R', X), \text{ if } c = 0 \\ (X, R'), \text{ if } c = 1 \end{cases}$
        $(Rel, \boldsymbol{C}) \overset{R}{\leftarrow} A_2^{O_2}(X', s_1, C^*)$
        $s_2 \leftarrow (Rel, X);$ return $(s_2, \boldsymbol{C})$

$\underline{B_3(s_2, \boldsymbol{K}) \text{ where } s_2 = (Rel, X)}$

        If $Rel(X, \boldsymbol{K})$, then $g \leftarrow 0$,
        otherwise $g \leftarrow 1;$ return $g$
```

Figure 5.7: PNM-ATK adversary B using CNM-ATK adversary Adv.

$$\mathtt{Adv}^{\mathsf{IND\text{-}PX\text{-}CY}}_{A,\Sigma'}(k) \equiv \Pr[\mathtt{Expt}^{\mathsf{IND\text{-}PX\text{-}CY}}_{A,\Sigma'}(k)] - \frac{1}{2}$$

where $\mathtt{Expt}^{\mathsf{IND\text{-}PX\text{-}CY}}_{A,\Sigma'}(k)$:

$$K \overset{\mathsf{U}}{\leftarrow} \{0,1\}^{l(k)}; (x_0, x_1, s) \overset{\mathsf{R}}{\leftarrow} A_1^{O_1, O_1'}(1^k)$$
$$b \overset{\mathsf{U}}{\leftarrow} \{0,1\}; y \overset{\mathsf{R}}{\leftarrow} \mathcal{E}'(K, x_b)$$
$$g \overset{\mathsf{R}}{\leftarrow} A_2^{O_2, O_2'}(1^k, s, y)$$
$$\text{return 1 iff} \quad g = b$$

and
If X = 0, then $O_1(\cdot) = \perp$ and $O_2(\cdot) = \perp$.
If X = 1, then $O_1(\cdot) = \mathcal{E}'(K, \cdot)$ and $O_2(\cdot) = \perp$.
If X = 2, then $O_1(\cdot) = \mathcal{E}'(K, \cdot)$ and $O_2(\cdot) = \mathcal{E}'(K, \cdot)$.
If Y = 0, then $O_1'(\cdot) = \perp$ and $O_2'(\cdot) = \perp$.
If Y = 1, then $O_1'(\cdot) = \mathcal{D}'(K, \cdot)$ and $O_2'(\cdot) = \perp$.
If Y = 2, then $O_1'(\cdot) = \mathcal{D}'(K, \cdot)$ and $O_2'(\cdot) = \mathcal{D}'(K, \cdot)$.

Figure 5.8: Advantage of IND-PX-CY-DEM

$$\mathtt{Adv}^{\textsf{NM-PX-CY}}_{A,\Sigma'}(k) \equiv \Pr[\mathtt{Expt}^{\textsf{NM-PX-CY}}_{A,\Sigma'}(k)=1] - \Pr[\widetilde{\mathtt{Expt}}^{\textsf{NM-PX-CY}}_{A,\Sigma'}(k)=1]$$

where $\mathtt{Expt}^{\textsf{NM-PX-CY}}_{A,\Sigma'}(k)$

$$\begin{gathered} K \leftarrow \{0,1\}^k; (M,s) \leftarrow A_1^{O_1,O_1'}(1^k) \\ x \leftarrow M; y \leftarrow \mathcal{E}'(K,x) \\ (R,\boldsymbol{y}) \leftarrow A_2^{O_2,O_2'}(s,y) \\ \boldsymbol{x} \leftarrow \mathcal{D}'(K,\boldsymbol{y}) \\ \text{return 1 iff } R(x,\boldsymbol{x}) \end{gathered}$$

$\widetilde{\mathtt{Expt}}^{\textsf{NM-PX-CY}}_{A,\Sigma'}(k)$

$$\begin{gathered} K \leftarrow \{0,1\}^k; (M,s) \leftarrow A_1^{O_1,O_1'}(1^k) \\ (x,\tilde{x}) \leftarrow M; \tilde{y} \leftarrow \mathcal{E}'(K,\tilde{x}) \\ (R,\widetilde{\boldsymbol{y}}) \leftarrow A_2^{O_2,O_2'}(s,\widetilde{y}) \\ \widetilde{\boldsymbol{x}} \leftarrow \mathcal{D}'(K,\widetilde{\boldsymbol{y}}) \\ \text{return 1 iff } R(x,\widetilde{\boldsymbol{x}}) \end{gathered}$$

and
If X = 0, then $O_1(\cdot) = \perp$ and $O_2(\cdot) = \perp$.
If X = 1, then $O_1(\cdot) = \mathcal{E}'(K,\cdot)$ and $O_2(\cdot) = \perp$.
If X = 2, then $O_1(\cdot) = \mathcal{E}'(K,\cdot)$ and $O_2(\cdot) = \mathcal{E}'(K,\cdot)$.
If Y = 0, then $O_1'(\cdot) = \perp$ and $O_2'(\cdot) = \perp$.
If Y = 1, then $O_1'(\cdot) = \mathcal{D}'(K,\cdot)$ and $O_2'(\cdot) = \perp$.
If Y = 2, then $O_1'(\cdot) = \mathcal{D}'(K,\cdot)$ and $O_2'(\cdot) = \mathcal{D}'(K,\cdot)$.

Figure 5.9: Advantage of NM-PX-CY-DEM

```
$B_1^{O_1,O'_1}(1^k)$
$(x_0, x_1, s) \leftarrow A^{O_1,O'_1}(1^k)$
$M = \{x_0, x_1\}$
return $(M, (x_0, x_1, s))$
```

```
$B_2^{O_2,O'_2}(1^k, (x_0, x_1, s), y)$
$b \leftarrow A^{O_2,O'_2}(1^k, s, y)$
choose $x'$ and semantic relation $R$
    where $R(x, x') = 1$ iff $x = x_b$
$y' \leftarrow \mathcal{E}'(K, x')$
return $(R, y')$
```

Figure 5.10: NM-P2-C2-DEM ⇒ IND-P2-C2-DEM

```
$B_1^{O_1,O'_1}(1^k)$
$(M, s) \leftarrow A_1^{O_1,O'_1}(1^k)$
$x_0, x_1 \leftarrow M$
return $(x_0, x_1, (x_0, s))$
```

```
$B_2^{O_2,O'_2}(1^k, (x_0, s), y)$
$(R, y) \leftarrow A_2^{O_2,O'_2}(1^k, s, y)$
$x \leftarrow \mathcal{D}'(K, y)$
if $(\perp \notin x \wedge R(x_0, x))$
return 0
else return 1
```

Figure 5.11: IND-P2-C2-DEM ⇒ NM-P2-C2-DEM

Chapter 6

UC KEM and UC DEM

6.1 UC KEM

Let $\Sigma = (\mathcal{G},\mathcal{E},\mathcal{D})$ be KEM. Hereafter, we define the KEM functionality $\mathcal{F}_{\text{KEM}}$ and protocol π_Σ that is constructed from KEM Σ and that has the same interface with the environment as $\mathcal{F}_{\text{KEM}}$.

Definition 20. *Let $\mathcal{F}_{\text{KEM}}$ be the KEM functionality shown in Fig.6.1, and let π_Σ be the KEM protocol in Fig.6.2.*

Here, note that there is no functionality of data transmission between parties in $\mathcal{F}_{\text{KEM}}$.

6.2 UC KEM Is Equivalent to IND-CCA2 KEM

This section shows that KEM Σ is UC secure if and only if Σ is IND-CCA2 (or NM-CCA2).

Theorem 6. *Let Σ be a KEM scheme, and $\mathcal{F}_{\text{KEM}}$ and π_Σ be as described in Definition 20. Protocol π_Σ UC-realizes $\mathcal{F}_{\text{KEM}}$ with respect to non-adaptive adversaries, if and only if Σ is IND-CCA2-KEM.*

Proof.
("Only if" part) Let $\Sigma = (\mathcal{G},\mathcal{E},\mathcal{D})$ be a KEM scheme. We prove that if Σ is

Functionality $\mathcal{F}_{\text{KEM}}$

$\mathcal{F}_{\text{KEM}}$ which runs with adversary Sim proceeds as follows:

Key Generation: Upon receiving (`KEM.KeyGen`, sid) from some party D, verify that sid=(D, sid') for some sid'. If not, then ignore the request. Else, hand (`KEM.KeyGen`, sid) to adversary Sim. Upon receiving (`Algorithms`, sid, e, d) from Sim where e and d are descriptions of PPT ITMs, output (`Encryption Algorithm`, sid, e) to D.

Encryption: Upon receiving (`KEM.Encrypt`, sid, e') from any party E, perform the following: If $e' \neq e$, or decryptor D is corrupted, then execute e' and obtain (K^*, C^*). Let $(key, cip) \leftarrow (K^*, C^*)$. Else, obtain (K^*, C^*) by e' and $R \overset{\cup}{\leftarrow} \{0,1\}^{l(k)}$, then let $(key, cip) \leftarrow (R, C^*)$ and record (key, cip). Output (`Key and Ciphertext`, sid, key, cip) to E.

Decryption: Upon receiving a value (`KEM.Decrypt`, sid, C^*) from D (and D only), perform the following: If there is recorded entry (K, C^*) for some K then return (`Shared Key`, sid, K) to D. Else, return (`Shared Key`, sid, $d(C^*)$) to D (If there is more than one K recorded for C^*, then output an error message).

Figure 6.1: KEM Functionality $\mathcal{F}_{\text{KEM}}$

not IND-CCA2-KEM, then π_Σ does not UC-realize $\mathcal{F}_{\text{KEM}}$. In more detail, we can construct environment Env such that, for any ideal process world adversary (simulator) Sim, Env can tell whether it is interacting with Adv and π_Σ or with Sim and the ideal protocol for $\mathcal{F}_{\text{KEM}}$, by using adversary G that breaks Σ in the sense of IND-CCA2-KEM with non-negligible advantage, i.e., $\mathtt{Adv}^{\text{IND-CCA2}}_{G,\Sigma}(k) > \mu(k)$).

Protocol π_Σ

π_Σ proceeds with parties E and D as follows:

Key Generation: Upon input (`KEM.KeyGen`, *sid*), party D verifies that $sid{=}(D, sid')$ for some sid'. If not, then ignore the request. Else, D obtains public key pk and secret key sk by executing algorithm $\mathcal{G}$, and generates $e \leftarrow \mathcal{E}(pk, \cdot)$ and $d \leftarrow \mathcal{D}(sk, \cdot)$, then outputs (`Encryption Algorithm`, *sid*, *e*).

Encryption: Upon input (`KEM.Encrypt`, *sid*, *e*), party E obtains pair $(key, cip) \leftarrow (K^*, C^*)$ of a key and a ciphertext by executing algorithm e and outputs (`Key and Ciphertext`, *sid*, key, cip).

Decryption: Upon input (`KEM.Decrypt`, *sid*, C^*), party D (that has d) obtains $K^* \leftarrow d(C^*)$ and outputs (`Shared Key`, *sid*, K^*).

Figure 6.2: KEM Protocol π_Σ

Env activates parties E and D, and uses adversary G as follows:

1. *Env* activates key decryptor D with (`KEM.KeyGen`, *sid*) for $sid{=}(D, 0)$, obtains encryption algorithm e, and hands e to G.

2. *Env* activates E with (`KEM.Encrypt`, *sid*, *e*), and obtains (key, cip). *Env* chooses $b \overset{\cup}{\leftarrow} \{0,1\}$ and $R \overset{\cup}{\leftarrow} \{0,1\}^{l(k)}$. If $b = 0$, then $key' \leftarrow key$. If $b = 1$, then $key' \leftarrow R$. *Env* hands (key', cip) to G as a target pair of key and ciphertext in the IND-CCA2 game shown in Fig. 2.1.

3. When G asks its decryption oracle to decrypt ciphertext $C^\dagger \neq cip$, *Env* activates D with input (`KEM.Decrypt`, $sid, C^\dagger$), obtains key $K^\dagger$, and hands $K^\dagger$ to G.

4. When G outputs $g \in \{0,1\}$, *Env* outputs $g \oplus b$ and halts.

Here note that *Env* corrupts no party and interacts with no adversary.

When *Env* interacts with π_Σ, the view of G interacting with *Env* is exactly the same as that behaving in the real IND-CCA2 game in Fig. 2.1. Therefore, in this case (say `Real`), $g = b$ with probability $> \frac{1}{2} + \mu(k)$.

In contrast, when *Env* interacts with the ideal process world for $\mathcal{F}_{\mathrm{KEM}}$, the view of G interacting with *Env* is independent of b, since b is independent of (key', cip) generated by *Env* in step 2 and is independent of the decryption result $K^\dagger$ in step 3 (as key' and $K^\dagger$ are random strings independent of b). Hence, in this case (say `Ideal`), $g = b$ with probability of exactly $\frac{1}{2}$.

Thus, $|\Pr[Env \to 0 \mid \texttt{Real}] - |\Pr[Env \to 0 \mid \texttt{Ideal}]| > \mu(k)$.

("If" part) We show that if π_Σ does not UC-realize $\mathcal{F}_{\mathrm{KEM}}$, then Σ is not IND-CCA2-KEM. To do so, we first assume that for any simulator *Sim* there exists real world adversary *Adv* and environment *Env* that distinguishes with probability $> \frac{1}{2} + \mu(k)$ whether it is interacting with *Sim* and the ideal process for $\mathcal{F}_{\mathrm{KEM}}$ or with *Adv* and π_Σ. We then show that there exists an IND-CCA2 attacker G against Σ using *Env*.

First we show that *Env* can distinguish (Adv, π_Σ) and $(Sim, \mathcal{F}_{\mathrm{KEM}})$ only when no party is corrupted. Since we are dealing with non-adaptive adversaries, there are three cases. Case 1: Sender E is corrupted (throughout the protocol). Case 2: Decryptor D is corrupted (throughout the protocol). Case 3: E and D are uncorrupted.

In Case 1, we can construct simulator *Sim* such that no *Env* can distinguish (Adv, π_Σ) and $(Sim, \mathcal{F}_{\mathrm{KEM}})$ as described hereafter.

1. When *Env* sends (`KEM.KeyGen`, *sid*) to D, D forwards it to $\mathcal{F}_{\mathrm{KEM}}$. $\mathcal{F}_{\mathrm{KEM}}$ sends (`KEM.KeyGen`, *sid*) to *Sim*, *Sim* computes (pk, sk) by executing algorithm $\mathcal{G}$, and generates e and d, where $e \leftarrow \mathcal{E}(pk, \cdot)$ and $d \leftarrow \mathcal{D}(sk, \cdot)$. *Sim* returns (`Algorithms`, *sid*, e, d) to $\mathcal{F}_{\mathrm{KEM}}$.

2. When *Env* sends (`KEM.Encrypt`, *sid*, e) to corrupted party E, i.e., *Sim*, *Sim* receives the message and sends it to the simulated copy of *Adv*, which replies to *Sim*. *Sim* then returns *Adv*'s reply (which may be $\perp$) to *Env*.

3. When Env sends (`KEM.Decrypt`, sid, C^*) to D, D forwards it to $\mathcal{F}_{\text{KEM}}$. $\mathcal{F}_{\text{KEM}}$ then returns (`Shared Key`, sid, $d(C^*)$), since E, i.e., Sim, sends no (`KEM.Encrypt`, sid, e) to $\mathcal{F}_{\text{KEM}}$, which records nothing as (key, cip). Note that, Sim does not receive any message in this step.

In this case, Env cannot distinguish (Adv, π_Σ) from $(Sim, \mathcal{F}_{\text{KEM}})$, because the message returned by Sim (using Adv) as E in the ideal world is the same as that returned by Adv as E in the real world, and (`Shared Key`, sid, $d(C^*)$) returned by $\mathcal{F}_{\text{KEM}}$ is exactly the same as that returned by D in the real world.

In Case 2, we can also construct simulator Sim such that no Env can distinguish (Adv, π_Σ) and $(Sim, \mathcal{F}_{\text{KEM}})$ as described hereafter.

1. When Env sends (`KEM.KeyGen`, sid) to the corrupted party D, i.e., Sim, receives the message and sends it to the simulated copy of Adv, which returns a reply message (which may be $\perp$) to Sim. Sim sends it to Env.

2. When Env sends (`KEM.Encrypt`, sid, e) to E, E forwards it to $\mathcal{F}_{\text{KEM}}$. $\mathcal{F}_{\text{KEM}}$ generates a corresponding pair (K^*, C^*) by executing e, sets $(key, cip) \leftarrow (K^*, C^*)$ and returns (`Key and Ciphertext`, sid, key, cip) to E, since D, i.e., Sim, sends no (`KEM.KeyGen`, sid) to $\mathcal{F}_{\text{KEM}}$, which records nothing as encryption algorithm e.

3. When Env sends (`KEM.Decrypt`, sid, C^*) to D, i.e., Sim, sends (`KEM.Decrypt`, sid, C^*) to Adv. Adv returns a reply (which may be $\perp$) to Sim, which forwards Adv's reply to Env.

In this case, Env cannot distinguish (Adv, π_Σ) and $(Sim, \mathcal{F}_{\text{KEM}})$ because the message returned by Sim (using Adv) as D in the ideal world is the same as that returned by Adv as D in the real world, and (`Key and Ciphertext`, sid, key, cip) returned by $\mathcal{F}_{\text{KEM}}$ is exactly the same as that returned by E in the real world.

Thus, Env cannot distinguish the real/ideal worlds in Cases 1 and 2. Hereafter, we consider only Case 3: E and D are uncorrupted.

Referring to the UC framework, three types of messages are sent from *Env* to *Adv*. The first message type is to corrupt either party, the second message type is to report on message sending, and the third message type is to deliver some message. In π_Σ, considered here, parties do not send messages to each other over the network. In addition, we consider the case that no party is corrupted. Therefore, there are no messages from *Env* to *Adv* (and *Sim*).

Since there exists at least one environment *Env* that can distinguish the real life world from the ideal process world for any simulator *Sim*, we consider the following special simulator *Sim*.

When *Sim* receives message (`KEM.KeyGen`, *sid*) from $\mathcal{F}_{\text{KEM}}$, *Sim* executes key generation algorithm $\mathcal{G}$ and obtains public key *pk* and secret key *sk*. *Sim* sets $e \leftarrow \mathcal{E}(pk,\cdot)$ and $d \leftarrow \mathcal{D}(sk,\cdot)$, and returns (`Algorithms`, *sid*, *e*, *d*) to $\mathcal{F}_{\text{KEM}}$.

We now show that we can construct adversary *G* that breaks IND-CCA2-KEM by using the simulated copy of *Env*, which distinguishes real/ideal worlds. To do so, we assume that there is environment *Env* such that

$$|\mathtt{IDEAL}_{\mathcal{F}_{\text{KEM}},Sim,Env}(k,z) - \mathtt{REAL}_{\pi_\Sigma,Adv,Env}(k,z)| > \mu(k).$$

We then show that *G* using *Env* can correctly guess *b* in the IND-CCA2 game in Fig. 2.1 with a probability of at least $\frac{1}{2} + \mu(k)/2\ell$, where ℓ is the total number of times the encryption oracle is invoked.

In the IND-CCA2 game, given target public-key (encryption algorithm) *e* and target pair (key, cip) from the encryption oracle with private random bit *b*, *G* is allowed to query the decryption oracle, and finally outputs *g*, which is *G*'s guess of *b*. *G* executes *Env* with the following simulated interaction as protocol $\pi_\Sigma/\mathcal{F}_{\text{KEM}}$.

G acts as indicated hereafter where K_i^*, C_i^* and R_i denote the *i*-th key, ciphertext and random value of the length $l(k)$, respectively.

1. When *Env* activates some party *D* with (`KEM.KeyGen`, *sid*), *G* lets *D* output (`Encryption Algorithms`, *sid*, *e*) where *e* is the target public-key (encryption algorithm) for *G* in the IND-CCA2 game.

2. For the first h times that *Env* asks some party E to generate (*key, cip*) with *sid*, G lets E return $(key, cip) \leftarrow (K_i^*, C_i^*)$ by using algorithm e.

3. The h-th time that *Env* asks to generate (*key, cip*) with *sid*, G queries its encryption oracle in the IND-CCA2 game, and obtains corresponding pair $(key, cip) \leftarrow (K_h^*, C_h^*)$ (when $b = 0$) or non-corresponding pair $(key, cip) \leftarrow (R_h, C_h^*)$ (when $b = 1$), where $R_h \overset{\cup}{\leftarrow} \{0,1\}^{l(k)}$. Accordingly, G hands pair (key, cip) to *Env*.

4. For the remaining $\ell - h$ times that *Env* asks E to generate (*key, cip*) with *sid*, G lets E return $(key, cip) \leftarrow (R_i, C_i^*)$, where $R_i \overset{\cup}{\leftarrow} \{0,1\}^{l(k)}$.

5. Whenever *Env* activates decryptor D with (`KEM.Decrypt`, *sid*, C^*), where $C^* = C_i^*$ for some i, G lets D return the corresponding key K_i^* or R_i^* for any i. If C^* is different from all C_i^*'s, G then poses C^* to its decryption oracle, obtains value v, and lets D return v to *Env*.

6. When *Env* halts, G outputs whatever *Env* outputs and halts.

We use a standard hybrid argument to analyze the success probability of G in the IND-CCA2 game.

For $h \in \{0, \dots, \ell\}$, let $\mathtt{Env}_h$ be an event that for the first h times that *Env* asks some party E to generate (*key, cip*) with *sid*, E returns $(key, cip) \leftarrow (K_i^*, C_i^*)$ by using algorithm e; the h-th time that *Env* asks E to generate (*key, cip*) with *sid*, E returns $(key, cip) \leftarrow (K_i^*, C_i^*)$ or $(key, cip) \leftarrow (R_i, C_i^*)$ where $R_i \overset{\cup}{\leftarrow} \{0,1\}^{l(k)}$. For the remaining $\ell - h$ times that *Env* asks E to generate (*key, cip*) with *sid*, E returns $(key, cip) \leftarrow (R_i, C_i^*)$ where $R_i \overset{\cup}{\leftarrow} \{0,1\}^{l(k)}$. The replies to *Env* from decryptor D are the same as those shown in step 5 above.

Let H_h be $\Pr[Env \to 1 | \mathtt{Env}_h]$. We then obtain the following inequality.

$$\sum_{h=1}^{\ell} |H_h - H_{h-1}| \geq |H_\ell - H_0|. \tag{6.1}$$

Here, from the construction of H_h it is clear that

$$H_0 = \mathtt{IDEAL}_{\mathcal{F}_{\mathrm{KEM}},Sim,Env}(k,z), \tag{6.2}$$

$$H_\ell = \mathtt{REAL}_{\pi_\Sigma,Adv,Env}(k,z). \tag{6.3}$$

Therefore,

$$\begin{aligned} &\sum_{h=1}^{\ell} |H_h - H_{h-1}| \geq |H_\ell - H_0| \\ = \quad & |\mathtt{REAL}_{\pi_\Sigma,Adv,Env}(k,z) - \mathtt{IDEAL}_{\mathcal{F}_{\mathrm{KEM}},Sim,Env}(k,z)| > \mu(k). \end{aligned}$$

Then there exists some $h \in \{1, \cdots \ell\}$ that satisfies

$$|H_h - H_{h-1}| \quad > \quad \mu(k)/\ell. \tag{6.4}$$

Here, w.l.o.g., let $H_{h-1} - H_h > \mu(k)/\ell$, since if $H_h - H_{h-1} > \mu(k)/\ell$ for Env, we can obtain $H_{h-1} - H_h > \mu(k)/\ell$ for Env^*, where Env^* outputs the opposite of Env's output bit.

In step 3 of G's construction, if G obtains the corresponding pair (K_h^*, C_h^*) (when $b = 0$), then the probability that Env outputs 1 is identical to H_h. On the other hand, if G obtains the non-corresponding pair of (R_h, C_h^*) (when $b = 1$), then the probability that Env outputs 1 is identical to H_{h-1}.

Since G's output follows Env's output,

$$H_h = \Pr[g = 1|b = 0] \text{ and} \tag{6.5}$$

$$H_{h-1} = \Pr[g = 1|b = 1], \tag{6.6}$$

where b is the private random bit of the encryption oracle in the IND-CCA2 game and g is G's output (G's guess of b).

Since $\Pr[g = 1|b = 0] + \Pr[g = 0|b = 0] = 1$, we obtain $\Pr[g = 0|b = 0] = 1 - \Pr[g = 1|b = 0]$.

Therefore, we obtain G's success probability,

$\Pr[\mathtt{Expt}^{\mathrm{IND-CCA2}}_{G,\Sigma}(k) = 1]$, as follows:

$$\begin{aligned}
&\Pr[\mathtt{Expt}^{\mathrm{IND-CCA2}}_{G,\Sigma}(k) = 1] \\
&\quad = \Pr[b = g] \\
&\quad = \Pr[b = 0] \times \Pr[g = 0|b = 0] \\
&\quad\quad + \Pr[b = 1] \times \Pr[g = 1|b = 1] \\
&\quad = \frac{1}{2} \times (\Pr[g = 0|b = 0] + \Pr[g = 1|b = 1]) \\
&\quad = \frac{1}{2} \times (1 - \Pr[g = 1|b = 0] + \Pr[g = 1|b = 1]) \\
&\quad = \frac{1}{2} \times (1 - H_h + H_{h-1}) > \frac{1}{2} + \mu(k)/2\ell.
\end{aligned}$$

That is, $\mathtt{Adv}^{\mathrm{IND-CCA2}}_{G,\Sigma}(k) > \mu(k)/2\ell$, which is not negligible in k since ℓ is polynomially bounded in k. □

6.3 UC DEM

Let $\Sigma' = (\mathcal{E}', \mathcal{D}')$ be a DEM scheme and let Σ'' be a $\mathcal{F}_{\mathrm{KEM}}$-hybrid DEM scheme. Hereafter we define the KEM and DEM functionality $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ and protocol $\pi_{\Sigma''}$ that is constructed from DEM Σ' in the $\mathcal{F}_{\mathrm{KEM}}$ hybrid model and that has the same interfaces that environment Env uses to communicate with $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$.

Definition 21. *Let $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ be the KEM and DEM, KEM-DEM, functionality shown in Fig.6.3 and in Fig.6.4, and let $\pi_{\Sigma''}$ be the KEM-DEM protocol in Fig.6.5 and Fig.6.6 .*

Here, note that there is no functionality for the data transmission between parties in $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$, and we consider that algorithm e in KEM.KeyGen of $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ outputs different key ciphertext C^*.

Functionality $\mathcal{F}_{\text{KEM-DEM}}$

$\mathcal{F}_{\text{KEM-DEM}}$ proceeds as follows, and is executed with party $P \in \{E_1, \ldots, E_n, D\}$ and adversary Sim.

KEM.KeyGen: Upon receiving (`KEM.KeyGen`, *sid*) from key decryptor D, verify that $sid{=}(D, sid')$ for some sid'. If not, then ignore the request. Else, hand (`KEM.KeyGen`, *sid*) to adversary Sim. Upon receiving (`Algorithms`, *sid*, e, d, e_{DEM}, d_{DEM}) from Sim, where e, d, e_{DEM} and d_{DEM} are descriptions of PPT TMs, output (`KEM Encryption Algorithm`, *sid*, e) to D.

KEM.Encrypt: Upon receiving (`KEM.Encrypt`, *sid*, e') from key encryptor $E_i (i \in \{1, \ldots, n\})$, perform the following:

- If $e' \neq e$, or key decryptor D is corrupted, then obtain K and C^* by e', record $(E_i, K, C^*, 0)$ and send (`KEM.Ciphertext`, *sid*, C^*) to E_i.
- Else, obtain C^* by e' and $K \overset{\cup}{\leftarrow} \{0,1\}^{l(k)}$, record $(E_i, K, C^*, 1)$ and send (`KEM.Ciphertext`, *sid*, C^*) to E_i.

KEM.Decrypt: Upon receiving (`KEM.Decrypt`, *sid*, C') from key decryptor D (and D only), perform the following:

- If C' is in the memory $(E_i, K, C', 1)$ for some E_i and K, record $(D, K, C', 1)$ and send *ok* to D.
- Else, record $(D, d(C'), C', 0)$ and send *ok* to D.

Figure 6.3: KEM-DEM Functionality (Part I)

Functionality $\mathcal{F}_{\text{KEM-DEM}}$

DEM.Encrypt: Upon receiving (`DEM.Encrypt`, *sid*, *m*, C') from party P, proceed as described below.

- If $(P, K, C', 1)$ is recorded in the memory for some K and $\tilde{P}$ is uncorrupted ($\tilde{P}$ denotes D if P is E_i, $\tilde{P}$ denotes $E_i^{C'}$ if P is D, where $E_i^{C'}$ denotes the party such that (E_i, *, C', 1) is recorded), then do as follows:
    1. Generate c by $e_{\text{DEM}}(K, \mu)$, where μ is a fixed message, and record (m, c, C') in the memory.
    2. Send (`DEM.Ciphertext`, *sid*, c) to P.
- Else if (($P, K, C', 1$) is recorded and $\tilde{P}$ is corrupted) or $(P, K, C', 0)$ is recorded in the memory for some K, then perform the following:
    1. Generate c using $e_{\text{DEM}}(K, m)$.
    2. Send (`DEM.Ciphertext`, *sid*, c) to P.
- Else, do nothing.

DEM.Decrypt: Upon receiving (`DEM.Decrypt`, *sid*, *c*, C') from party P, proceed as follows:

- If $(P, K, C', 1)$ is recorded in the memory for some K and (m, c, C') is recorded, then send (`DEM.Plaintext`, *sid*, m) to P.
- Else if $(P, K, C', *)$ is recorded in the memory for some K, send (`DEM.Plaintext`, *sid*, $d_{\text{DEM}}(K, c)$) to P.
- Else, do nothing.

Figure 6.4: KEM-DEM Functionality (Part II)

Protocol $\pi_{\Sigma''}$

$\pi_{\Sigma''}$ proceeds as follows, and is executed with party $P \in \{E_1, \ldots, E_n, D\}$ and an ideal functionality $\mathcal{F}_{\text{KEM}}$.

KEM.KeyGen: Upon input (`KEM.KeyGen`, *sid*) within key decryptor D,

1. D sends (`KEM.KeyGen`, sid') to $\mathcal{F}_{\text{KEM}}$.
2. Upon receiving (`KEM Key`, sid', e) from $\mathcal{F}_{\text{KEM}}$, D outputs (`KEM Encryption Algorithm`, *sid*, e).

KEM.Encrypt: Upon input (`KEM.Encrypt`, *sid*, e') within key encryptor E_i,

1. E_i sends (`KEM.Encrypt`, sid', e') to $\mathcal{F}_{\text{KEM}}$.
2. Upon receiving (`Key and Ciphertext`, sid', K, C^*) from $\mathcal{F}_{\text{KEM}}$, E_i stores (K, C^*) in it's memory.
3. E_i outputs (`KEM.Ciphertext`, *sid*, C^*).

KEM.Decrypt: Upon input (`KEM.Decrypt`, *sid*, C') within D,

1. D sends (`KEM.Decrypt`, sid', C^*) to $\mathcal{F}_{\text{KEM}}$.
2. Upon receiving (`Shared Key`, sid', K), D stores (K, C') in it's memory.
3. D outputs *ok*.

Figure 6.5: KEM-DEM Protocol (Part I)

The revised point from the previous definition [41, 42] is to remove the restriction that the previous $\mathcal{F}_{\text{KEM-DEM}}$ can have only one key in the DEM phase. To solve this problem, we made the current functionality accept the multiple key ciphertexts generated by (DEM.Decrypt, sid, c, C') in DEM.Decrypt of $\mathcal{F}_{\text{KEM-DEM}}$ where c is the ciphertext of a message and C' is the encryption of some key.

Protocol $\pi_{\Sigma''}$

DEM.Encrypt: Upon input ($\mathtt{DEM.Encrypt}$, *sid*, m, C') within party P, proceed as described below.

- If (K, C') exists in P's memory, P obtains ciphertext $c = \mathcal{E}'(K, m)$ and outputs ($\mathtt{DEM.Ciphertext}$, *sid*, c).
- Else, do nothing.

DEM.Decrypt: Upon input ($\mathtt{DEM.Decrypt}$, *sid*, c, C') within party P, proceed as described below.

- If (K, C') exists in P's memory, P obtains message $m = \mathcal{D}'(K, c)$ and outputs ($\mathtt{DEM.Plaintext}$, *sid*, m).
- Else, do nothing.

Figure 6.6: KEM-DEM Protocol (Part II)

6.4 UC DEM Is Equivalent to IND-P2-C2 DEM

The following theorem implies that UC DEM is equivalent to IND-P2-C2 DEM.

Theorem 7. *Protocol $\pi_{\Sigma''}$ UC-realizes $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ with respect to non-adaptive adversaries in the $\mathcal{F}_{\mathrm{KEM}}$-hybrid model, if and only if $\pi_{\Sigma''}$ is IND-P2-C2-DEM.*

Proof.
("only if" part) We prove that if $\pi_{\Sigma''}$ is not IND-P2-C2-DEM secure, then $\pi_{\Sigma''}$ does not UC-realize $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$. More specifically, we can construct an environment, *Env*, such that for any ideal process world adversary (simulator) Sim, *Env* can tell whether it is interacting with *Adv* and $\pi_{\Sigma''}$ in the $\mathcal{F}_{\mathrm{KEM}}$ hybrid model or with Sim and the ideal protocol for $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ by using adversary F that breaks IND-P2-C2-DEM with non-negligible advantage, i.e., $\mathtt{Adv}^{\mathrm{IND-P2-C2}}_{F,\Sigma''}(k) > \mu(k)$).

Env activates party E_i and D, and uses adversary F as follows:

1. *Env* activates key receiver D with (`KEM.KeyGen`, *sid*) for $sid = (D, 0)$ and obtains encryption algorithm e.

2. *Env* activates key encryptor E_i with (`KEM.Encrypt`, *sid*, e') and obtains C^* from the output (`KEM.Ciphertext`, *sid*, C^*).

3. *Env* activates D with (`KEM.Decrypt`, *sid*, C^*) and obtains *ok*.

4. When F generates two plaintexts (m_0, m_1), *Env* chooses $b \overset{\cup}{\leftarrow} \{0,1\}$, activates E_i with (`DEM.Encrypt`, *sid*, m_b, C^*), and then obtains c from the output (`DEM.Ciphertext`, *sid*, c). *Env* hands c to F in the IND-P2-C2-DEM game shown in Fig. 5.8.

5. When F asks its encryption oracle to encrypt message $m^\flat$ (which may be m_0 or m_1), *Env* activates E_i with input (`DEM.Encrypt`, *sid*, $m^\flat$, C^*), obtains ciphertext $c^\flat$, and hands $c^\flat$ to F.

6. When F asks its decryption oracle to decrypt ciphertext $c^\dagger \neq c$, *Env* activates D with input (`DEM.Decrypt`, *sid*, $c^\dagger$, C^*), obtains message $m^\dagger$, and hands $m^\dagger$ to F.

7. When F outputs $g \in \{0,1\}$, *Env* outputs $g \oplus b$ and halts.

Here note that *Env* corrupts no party and interacts with no adversary.

When *Env* interacts with $\pi_{\Sigma''}$, the view of F interacting with *Env* is exactly the same as that behaving in the real IND-P2-C2 game in Fig. 5.8. Therefore, in this case (say `Real`), $g = b$ with probability $> \frac{1}{2} + \mu(k)$.

In contrast, when *Env* interacts with the ideal process world for $\mathcal{F}_{\text{KEM-DEM}}$, the view of F interacting with *Env* is independent of b, since b is independent of (m_0, m_1, c, μ) in step 4, and is independent of the encryption and decryption result $c^\flat$ and $m^\dagger$ in steps 5 and 6 (since $c^\flat$, m_0, m_1 and $m^\dagger$ are random strings independent of b). Hence, in this case (say `Ideal`), $g = b$ with probability of exactly $\frac{1}{2}$.

Thus, $|\Pr[Env \to 0 \mid \texttt{Real}] - |\Pr[Env \to 0 \mid \texttt{Ideal}]| > \mu(k)$.

("if" part) We show that if $\pi_{\Sigma''}$ does not UC-realize $\mathcal{F}_{\text{KEM-DEM}}$ in the $\mathcal{F}_{\text{KEM}}$-hybrid model, then $\pi_{\Sigma''}$ is not IND-P2-C2-DEM. To do so, we first assume that for any simulator Sim there is adversary Adv and environment Env that distinguishes with probability $> \frac{1}{2} + \mu(k)$ whether it interacts with Sim and $\mathcal{F}_{\text{KEM-DEM}}$ or with Adv and $\pi_{\Sigma''}$. We then show that there exists an IND-P2-C2-DEM attacker F against Σ'' using Env in the $\mathcal{F}_{\text{KEM}}$-hybrid model.

First we show that Env can distinguish $(Adv, \pi_{\Sigma''})$ in the $\mathcal{F}_{\text{KEM}}$-hybrid model and $(Sim, \mathcal{F}_{\text{KEM-DEM}})$ only when no party is corrupted. Since we are dealing with non-adaptive adversaries, there are three cases. Case 1: Sender E_i is corrupted (throughout the protocol). Case 2: Decryptor D is corrupted (throughout the protocol). Case 3: E_i and D are uncorrupted.

These cases are dealt with using the $\mathcal{F}_{\text{KEM}}$-hybrid model, so Env cannot tell whether Env interacts with protocol $\pi_{\Sigma''}$ or ideal $\mathcal{F}_{\text{KEM-DEM}}$ in the KEM= $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ phase. The KEM phase in all cases is performed as described hereafter.

1. When Env sends (KEM.KeyGen, sid) to D, $\mathcal{F}_{\text{KEM-DEM}}$ sends (KEM.KeyGen, sid) to Sim, Sim computes (pk, sk) by executing algorithm $\mathcal{G}$, and generates e, d, e_{DEM} and d_{DEM} where $e \leftarrow \mathcal{E}(pk, \cdot)$, $d \leftarrow \mathcal{D}(sk, \cdot)$, $e_{\text{DEM}} \leftarrow \mathcal{E}'$ and $d_{\text{DEM}} \leftarrow \mathcal{D}'$. Sim returns (Algorithms, sid, e, d, e_{DEM}, d_{DEM}) to $\mathcal{F}_{\text{KEM-DEM}}$ and $\mathcal{F}_{\text{KEM-DEM}}$ forwards (KEM Encryption Algorithm, sid, e) to D.

2. When Env sends (KEM.Encrypt, sid, e) to corrupted party E_i, E_i receives output (KEM.Ciphertext, sid, C^*).

3. When Env sends (KEM.Decrypt, sid, C^*) to D, D receives output ok.

We assume that Env cannot distinguish the ideal/real world in the KEM phase of all cases (hereafter, we discuss all cases after the KEM phase is finished).

In Case 1, we can construct simulator Sim such that no Env can distinguish $(Adv, \pi_{\Sigma''})$ in the $\mathcal{F}_{\text{KEM}}$-hybrid model and $(Sim, \mathcal{F}_{\text{KEM-DEM}})$ as described hereafter.

1. When Env sends (`DEM.Encrypt`, sid, m, C^*) to corrupted party E_i, i.e., Sim, Sim receives the message and sends it to the simulated copy of Adv, which replies to Sim. Sim then returns Adv's reply (which may be $\perp$) to Env.

2. When Env sends (`DEM.Decrypt`, sid, c, C^*) to D, D forwards it to $\mathcal{F}_{\text{KEM-DEM}}$. $\mathcal{F}_{\text{KEM-DEM}}$ then returns (`DEM.Plaintext`, sid, $d_{\text{DEM}}(K, c)$) since E, i.e., Sim, sends no (`DEM.Encrypt`, sid, m, C') to $\mathcal{F}_{\text{KEM-DEM}}$, which records nothing as (m, c, C'). Note that, Sim does not receive any message in this step.

In this case, Env cannot distinguish $(Adv, \pi_{\Sigma''})$ and $(Sim, \mathcal{F}_{\text{KEM}})$ because the message returned by Sim as E_i in the ideal world is the same as that returned by Adv as E_i in the real world, and (`DEM.Plaintext`, sid, $d_{\text{DEM}}(K, c)$) returned by $\mathcal{F}_{\text{KEM-DEM}}$ is exactly the same as that returned by D in the real world.

In Case 2, we can also construct simulator Sim such that no Env can distinguish $(Adv, \pi_{\Sigma''})$ and $(Sim, \mathcal{F}_{\text{KEM-DEM}})$ as described hereafter.

1. When Env sends (`DEM.Encrypt`, sid, m, C^*) to E_i, E_i forwards it to $\mathcal{F}_{\text{KEM-DEM}}$. $\mathcal{F}_{\text{KEM-DEM}}$ generates c using $e_{\text{DEM}}(K, m)$ and returns (`DEM.Ciphertext`, sid, c) to P to E_i, since D, i.e., Sim, is corrupted by Adv, which records nothing as ciphertext c.

2. When Env sends (`DEM.Decrypt`, sid, c, C^*) to D, i.e., Sim, Sim sends it to Adv. Adv returns a reply (which may be $\perp$) to Sim, which forwards Adv's reply to Env.

In this case, Env cannot distinguish $(Adv, \pi_{\Sigma''})$ from $(Sim, \mathcal{F}_{\text{KEM-DEM}})$ because the message returned by Sim (using Adv) as D in the ideal world is the same as that returned by Adv as D in the real world, and (`DEM.Decrypt`, sid, c, C^*) returned by $\mathcal{F}_{\text{KEM-DEM}}$ is exactly the same as that returned by E_i in the real world.

Thus, Env cannot distinguish the real/ideal worlds in Cases 1 and 2. Hereafter, we consider only Case 3: E_i and D are uncorrupted.

Referring to the UC framework, three types of messages are sent from *Env* to *Adv*. The first message type is to corrupt either party, the second message type is to report on message sending, and the third message type is to deliver some message. In protocol $\pi_{\Sigma''}$ considered here, parties do not send messages to each other over the network. In addition, we consider the case that no party is corrupted. Therefore, there are no messages from *Env* to *Adv* (and *Sim*).

Since there exists at least one environment *Env* that can distinguish the real life world from the ideal process world for any simulator *Sim*, we consider the following special simulator *Sim*.

- Receiving (KEM.KeyGen, *sid*) from $\mathcal{F}_{\text{KEM-DEM}}$, *Sim* obtains (*pk*, *sk*) by executing F and sets KEM encryption algorithm $e \leftarrow \mathcal{E}(pk)$, the KEM decryption algorithm, and $d \leftarrow \mathcal{D}(sk, \cdot)$. *Sim* then chooses DEM encryption algorithm $e_{\text{DEM}} \leftarrow \mathcal{E}'$ and DEM decryption algorithm $d_{\text{DEM}} \leftarrow \mathcal{D}'$ and sends (Algorithms, *sid*, *e*, *d*, e_{DEM}, d_{DEM}) to $\mathcal{F}_{\text{KEM-DEM}}$.

We now show that we can construct adversary F that breaks IND-P2-C2-DEM by using the simulated copy of *Env* which distinguishes real/ideal worlds in the $\mathcal{F}_{\text{KEM}}$-hybrid model. To do so, we assume that there is an environment *Env* such that

$$|\mathtt{IDEAL}_{\mathcal{F}_{\text{KEM-DEM}},Sim,Env}(k,z) - \mathtt{REAL}_{\pi_{\Sigma''},Adv,Env}(k,z)| > \mu(k),$$

when *Env* communicates with the message sending party $E_i \in \{E_1, \cdots, E_n\}$ and the message receiving party D.

We then show that F using *Env* can correctly guess b in the IND-P2-C2 game in Fig.5.8 with the probability of at least $\frac{1}{2} + \mu(k)/2n\ell$, where ℓ is the total number of times the encryption oracle is invoked and n is the number of all message sending parties E_i ($i \in \{1, \cdots, n\}$).

In the IND-P2-C2 game, F chooses a target message pair (x_0, x_1) with $|x_0| = |x_1|$. Given ciphertext y with private random bit $b \overset{\cup}{\leftarrow} \{0,1\}$ selected by the encryption oracle, F is allowed to query the encryption and decryption oracles, and finally outputs g, which is F's guess of b. F executes Env with the following simulated interaction as protocol $\pi_{\Sigma''}/\mathcal{F}_{\text{KEM-DEM}}$ in the $\mathcal{F}_{\text{KEM}}$-hybrid model.

F performs as described hereafter where k, ℓ, m_j, c_j, K_i, C_i, K_{atk} and C_{atk} denote the security parameter, the total number of encrypting messages that Env activates some party E_i with DEM.Encrypt, the j-th message, the j-th ciphertext, the key of F' choosing for message sending party E_i, the ciphertext of key for E_i, the shared key gained by using $\mathcal{F}_{\text{KEM}}$ between the message sending party E_{atk} and the message receiving party D, and the key ciphertext of K_{atk}, respectively. For some $h \in \{0, \cdots, \ell\}$,

1. F randomly selects one party E_{atk}.

2. For the first h times that Env activates some party E_i with ($\texttt{DEM.Encrypt}$, sid, m_j, C_i) to encrypt some message m_j, if $E_i \neq E_{atk}$, F lets E_i return $c_j \overset{\text{R}}{\leftarrow} e_{\text{DEM}}(K_i, m_j)$, where $K_i \overset{\text{U}}{\leftarrow} \{0,1\}^{l(k)}$ is F's chosen key for party E_i. Else, i.e., $E_i = E_{atk}$, and F lets E_{atk} return c_j after asking F's encryption oracle regarding m_j.

3. The h-th time that Env activates E_i with ($\texttt{DEM.Encrypt}$, sid, m_h, C_{atk}), if $E_i \neq E_{atk}$, F halts. Else, i.e., $E_i = E_{atk}$, then F queries its encryption oracle regarding $(x_0, x_1) \leftarrow (m_h, \mu)$ in the IND-P2-C2 game, and obtains corresponding ciphertext $c_h \leftarrow e_{\text{DEM}}(K_{atk}, m_h)$ (when $b = 0$) or non-corresponding ciphertext $c_h \leftarrow e_{\text{DEM}}(K_{atk}, \mu)$ (when $b = 1$). F lets E_{atk} return c_h to Env.

4. For the remaining $\ell - h$ times that Env activates some party E_i with ($\texttt{DEM.Encrypt}$, sid, m_j, C_i) to encrypt some message m_j, if $E_i \neq E_{atk}$, F lets E_i return $c_j \overset{\text{R}}{\leftarrow} e_{\text{DEM}}(K_i, \mu)$, where μ is the fixed message. Else, i.e., $E_i = E_{atk}$, then F lets E_{atk} return c_j after asking F's encryption oracle regarding μ.

5. Whenever Env activates D with ($\texttt{DEM.Decrypt}$, sid, c, C_i) where $c = c_j$ for some j, F lets D return the corresponding message m_j. Here, if

c is not all c_j, then F generates the decryption message of c_j with the key K_i for C_i and lets D return it to Env. Here, if $C_i = C_{atk}$, then F asks to its decryption oracle regarding c_j, obtains value v, and lets D return v to Env.

6. When Env halts, F outputs whatever Env outputs and halts.

Here, we also use a standard hybrid argument to analyze success probability of F in the IND-P2-C2 game.

For $h \in \{0, \dots, \ell\}$, let $\mathtt{Env}_h$ be an event that for the first h times that Env asks some party E_i (which may be E_{atk}) to generate ciphertext c_j with sid, E_i returns m_j's encryption c_j according to the above mentioned ways. For the h-th time that Env asks E_i (which may be E_{atk}) to generate ciphertext c_j with sid, E_i returns m_j's encryption or μ's encryption and for the remaining $\ell - h$ times that Env asks E_i (which may be E_{atk}) to generate c_j with sid, E_i returns μ's encryption c_j. The replies to Env from decryptor D are the same as those shown in step 5 above.

Let H_h be $\Pr[Env \to 1 | \mathtt{Env}_h]$. We then obtain the following inequality.

$$\sum_{h=1}^{\ell} |H_h - H_{h-1}| \geq |H_\ell - H_0|. \tag{6.7}$$

Here, from the construction of H_h it is clear that

$$H_0 = \mathtt{IDEAL}_{\mathcal{F}_{\text{KEM-DEM}}, Sim, Env}(k, z) \quad \text{and} \tag{6.8}$$

$$H_\ell = \mathtt{REAL}_{\pi_{\Sigma''}, Adv, Env}(k, z). \tag{6.9}$$

Therefore,

$$\begin{aligned}\sum_{h=1}^{\ell} |H_h - H_{h-1}| &\geq |H_\ell - H_0| \\ &= |\mathtt{REAL}_{\pi_{\Sigma''},Adv,Env}(k,z) - \mathtt{IDEAL}_{\mathcal{F}_{\text{KEM-DEM}},Sim,Env}(k,z)| \\ &> \mu(k).\end{aligned} \tag{6.10}$$

Then there exists some $h \in \{1, \cdots \ell\}$ that satisfies

$$|H_h - H_{h-1}| \quad > \quad \mu(k)/\ell. \tag{6.11}$$

Here, w.l.o.g., let $H_{h-1} - H_h > \mu(k)/\ell$, since if $H_h - H_{h-1} > \mu(k)/\ell$ for Env, we can obtain $H_{h-1} - H_h > \mu(k)/\ell$ for Env^*, where Env^* outputs the opposite of Env's output bit.

In step 3 of F's construction, F can continue the IND-P2-C2-DEM game, when the h-th time activation occurs on just E_{atk}. The probability that Env activates E_{atk} from all parties $E_i \in \{E_0, \cdots, E_n\}$ is $1/n$. If F obtains the corresponding pair of $(m_h,\ c_h)$ (when $b = 0$), then the probability that Env outputs 1 is identical to H_h/n. On the other hand, if F obtains the non-corresponding ciphertext of $(\mu,\ c_j)$ (when $b = 1$), then the probability that Env outputs 1 is identical to H_{h-1}/n.

Since F's output follows Env's output,

$$\Pr[g = 1|b = 0] = H_h/n \ \text{ and} \tag{6.12}$$

$$\Pr[g = 1|b = 1] = H_{h-1}/n, \tag{6.13}$$

where b is the private random bit of the encryption oracle in the IND-P2-C2 game and g is F's output (F's guess of b).

Since $\Pr[g = 1|b = 0] + \Pr[g = 0|b = 0] = 1$, we obtain $\Pr[g = 0|b = 0] = 1 - \Pr[g = 1|b = 0]$.

Therefore, from the above equalities, we obtain F's success probability, $\Pr[\mathtt{Expt}^{\text{IND-P2-C2}}_{F,\Sigma''}(k) = 1]$, as follows:

$$\begin{aligned}
\Pr[\mathtt{Expt}^{\mathrm{IND-P2-C2}}_{F,\Sigma''}(k)=1] &= \Pr[b=g] \\
&= \Pr[b=0]\times\Pr[g=0|b=0] \\
&\quad +\Pr[b=1]\times\Pr[g=1|b=1] \\
&= \frac{1}{2}\times(\Pr[g=0|b=0]+\Pr[g=1|b=1]) \\
&= \frac{1}{2}\times(1-\Pr[g=1|b=0]+\Pr[g=1|b=1]) \\
&= \frac{1}{2}\times(1-\frac{H_h}{n}+\frac{H_{h-1}}{n}) > \frac{1}{2}+\mu(k)/2n\ell.
\end{aligned}$$

That is, $\mathtt{Adv}^{\mathrm{IND-P2-C2}}_{F,\Sigma''}(k) > \mu(k)/2n\ell$, which is not negligible in k since n and ℓ are polynomially bounded in k.

Finally, we conclude that if $\pi_{\Sigma''}$ does not UC-realize $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ in the $\mathcal{F}_{\mathrm{KEM}}$-hybrid model, then $\pi_{\Sigma''}$ is not IND-P2-C2-DEM. □

Here, we define protocol $\pi_{\Sigma'}$ and obtain Theorem 8.

Theorem 8. *Let $\Sigma' = (\mathcal{E}', \mathcal{D}')$ be a DEM scheme. Let Σ'' be a $\mathcal{F}_{\mathrm{KEM}}$-hybrid Σ'. Protocol $\pi_{\Sigma''}$ is IND-P2-C2-DEM if and only if $\pi_{\Sigma'}$ is IND-P2-C2-DEM (or Σ' is IND-P2-C2 DEM).*

Proof. **("only if" part)** We show that if $\pi_{\Sigma'}$ is not IND-P2-C2-DEM then $\pi_{\Sigma''}$ is not IND-P2-C2-DEM. From the definition of $\pi_{\Sigma''}$ and the fact that $\pi_{\Sigma''}$ is in the $\mathcal{F}_{\mathrm{KEM}}$-hybrid model, this is trivial.

("if" part) We show that if $\pi_{\Sigma''}$ is not IND-P2-C2-DEM then $\pi_{\Sigma'}$ is not IND-P2-C2-DEM. This is also trivial from the definition of $\pi_{\Sigma'}$. □

Theorem 9. *Protocol $\pi_{\Sigma''}$ UC-realizes $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ with respect to non-adaptive adversaries in the $\mathcal{F}_{\mathrm{KEM}}$-hybrid model, if and only if Σ' is IND-P2-C2 DEM.*

Proof. This is also trivial from Theorem 7 and Theorem 8. □

From Theorems 7, 8, and 9, we obtain that UC DEM is equivalent to IND-P2-C2 DEM.

Chapter 7

Three Cryptographic Channels

7.1 Three Cryptographic Channels

In this section, we introduce three cryptographic channels: the SC, 2AC, and DIC.

7.1.1 SC

A SC is a channel such that the initiator (message sender) and the receiver (message receiver) can safely transmit messages to each other without the content being retrieved by a third party or adversary. This SC consists of three sessions: the establish session, data sending session, and expire session. 1. In the establish session a session is created between the initiator and the receiver to prepare for sending the message. 2. In the data sending session a message is safely sent to the message receiver. 3. In the expire session the existing session is terminated.

Definition 22. *The SC functionality, $\mathcal{F}_{\mathrm{SC}}$, is defined in Fig. 7.1 and the code for SC functionality,* F_{SC}*, is defined in Fig. 7.2 and Fig.7.3. ($\overline{X}$ ($X \in \{\mathrm{Init}, \mathrm{Rec}\}$) means that if $X = \mathrm{Init}$, then $\overline{X} = \mathrm{Rec}$, else if $X = \mathrm{Rec}$ then $\overline{X} = \mathrm{Init}$).*

7.1.2 2AC

An AC is one of the three cryptographic channels and is used to send some messages to the receiver from unknown senders (”anonymously”). The adversary can identify the receiver and read the message content, but cannot identify who sent the message to the receiver. When two senders and a receiver anonymously communicate using this channel, we say the channel is a 2AC. That is, one of the two senders sends a message to the receiver. Note that the 2AC can also be used when the receiver and one of the senders is the same process.

Definition 23. *The 2AC functionality,* $\mathcal{F}_{\mathrm{2AC}}$*, is defined in Fig. 7.4 and the code for the 2AC functionality,* $\mathrm{F}_{\mathrm{2AC}}$*, is defined in Fig. 7.5. and Fig. 7.6.*

7.1.3 DIC

A DIC is one of the three cryptographic channels. A DIC can be used to send some messages from the initiator to the receiver in a direction-indeterminable manner. An adversary can read the transmitted messages, but cannot identify the sender (and the receiver), i.e., the direction of the message transmission is indeterminable.

Definition 24. *The DIC functionality,* $\mathcal{F}_{\mathrm{DIC}}$*, is defined in Fig. 7.7 and the code for DIC functionality,* $\mathrm{F}_{\mathrm{DIC}}$*, is defined in Fig. 7.8 and Fig.7.9.*

Functionality $\mathcal{F}_{\text{SC}}$

$\mathcal{F}_{\text{SC}}$ proceeds as follows, running with parties $P \in \{\text{Init}, \text{Rec}\}$ and an adversary.

Establish Session:

Upon receiving ($\texttt{Establish}_{\text{SC}}$, $\texttt{sid}_{\text{SC}}$) from some party Init, verifies that $\texttt{sid}_{\text{SC}} = (\text{Init}, \text{Rec}, \texttt{sid}_{\text{SC}}{}')$ for Rec, then sends ($\texttt{SID}, \texttt{sid}_{\text{SC}}$) to the adversary, and waits to receive ($\texttt{Establish}_{\text{SC}}$, $\texttt{sid}_{\text{SC}}$) from Rec. Upon receiving this value, sets a boolean variable as active.

Data Sending Session:

Upon receiving ($\texttt{Send}$, $\texttt{sid}_{\text{SC}}, m$) from some party P, and if active is set, sends ($\texttt{Send}, \texttt{sid}_{\text{SC}}, |m|$) to the adversary. Upon receiving ($\texttt{Response}, \texttt{sid}_{\text{SC}}, ok$) from the adversary, sends ($\texttt{Receive}, \texttt{sid}_{\text{SC}}, m$) to the other party $\overline{P}$.

Expire Session:

Upon receiving ($\texttt{Expire}_{\text{SC}}$, $\texttt{sid}_{\text{SC}}$) from either party, unsets the variable active.

Figure 7.1: Secure Channel Functionality $\mathcal{F}_{\text{SC}}$

Code for Secure Channel Functionality, F_{SC}, where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Signature:

```
    $\mathtt{sid}_{\mathrm{SC}} = (\mathrm{Init}, \mathrm{Rec}, \mathtt{sid}'_{\mathrm{SC}})$
      Input:                                                        Output:
      $\mathit{receive}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$           $\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$
      $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_X$                 $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}}$
      $\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, ok)_{\mathrm{Adv}}$        $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, mes)_{\overline{X}}$
      $\mathit{receive}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$              $\mathit{send}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$
```

State:

```
      $estcond_X \in \{\bot, \top\}$, initially $\bot$            $active \in \{\bot, \top\}$, initially $\bot$
      $okcond_{\mathrm{Adv}} \in \{\bot, \top\}$, initially $\bot$          $mes \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$
      $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$
```

Tasks:

```
    $\{\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}, \mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}},$
    $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, mes)_{\overline{X}}, \mathit{send}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}\}$
```

Figure 7.2: Code for Secure Channel Functionality $\mathcal{F}_{\mathrm{SC}}$, F_{SC} (Part I)

Code for Secure Channel Functionality, F_{SC}, where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Transitions:

Establish Session:

ESS1. $\mathit{receive}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$
pre: $\mathit{active}, \mathit{ntask} = \bot$
eff: $\mathit{estcond}_X := \top$
If $\mathit{estcond}_X = \top$ for all X then $\mathit{active} := \top$ and $\mathit{ntask} :=$ ESS2.
Else do nothing.

ESS2. $\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$
pre:$\mathit{active} = \top$ and $\mathit{ntask} =$ ESS2
eff:$\mathit{ntask} := \bot$

Data Sending Session:

DSS1. $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_X$
pre:$\mathit{active} = \top$ and $\mathit{mes}, \mathit{ntask} = \bot$
eff: $\mathit{mes} := m$ and $\mathit{ntask} :=$ DSS2

DSS2. $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}}$
pre: $\mathit{okcond}_{\mathrm{Adv}} = \bot$, $m := \mathit{mes}$ and $\mathit{ntask} =$ DSS2
eff: $\mathit{ntask} :=$ DSS3

DSS3. $\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, \mathit{ok})_{\mathrm{Adv}}$
pre:$\mathit{ntask} =$ DSS3
eff: $\mathit{okcond}_{\mathrm{Adv}} := \top$ and $\mathit{ntask} :=$ DSS4

DSS4. $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{mes})_{\overline{X}}$
pre: $\mathit{ntask} =$ DSS4
eff: mes, $\mathit{okcond}_{\mathrm{Adv}}$, and $\mathit{ntask} := \bot$

Expire Session:

EXS1. $\mathit{receive}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$
pre: $\mathit{active} \neq \bot$ and $\mathit{mes} = \bot$
eff: $\mathit{ntask} :=$ EXS2

EXS2. $\mathit{send}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$
pre: $\mathit{ntask} =$ EXS2
eff: $\mathit{active}, \mathit{ntask}$ and $\mathit{estcond}_X := \bot$ for all X

Figure 7.3: Code for Secure Channel Functionality $\mathcal{F}_{\mathrm{SC}}$, F_{SC} (Part II)

Functionality $\mathcal{F}_{2AC}$

$\mathcal{F}_{2AC}$ proceeds as follows, running with party $P \in \{\text{Init}_i, \text{Rec}\}$ ($i \in \{1,2\}$) and an adversary. Here, Init_1 and Init_2 are the message sending party and Rec is the message receiving party.

Establish Session:

Upon receiving ($\texttt{Establish}_{\texttt{2AC}}$, $\texttt{sid}_{\texttt{2AC}}$) from message sending party Init_i ($i \in \{1,2\}$), verifies that $\texttt{sid}_{\texttt{2AC}} = (\{\text{Init}_1, \text{Init}_2\}, \text{Rec}, \texttt{sid}_{\texttt{2AC}}{}')$, sends ($\texttt{SID}, \texttt{sid}_{\texttt{2AC}}$) to the adversary, and waits to receive ($\texttt{Establish}_{\texttt{2AC}}$, $\texttt{sid}_{\texttt{2AC}}$) from other party $\text{Init}_{\bar{i}}$. If message from all party received, sets a boolean value as active.

Data Sending Session:

Upon receiving ($\texttt{Send}$, $\texttt{sid}_{\texttt{2AC}}$, m) from message sending party Init_i, and if active is set, sends ($\texttt{Send}, \texttt{sid}_{\texttt{2AC}}$, m) to the adversary. Upon receiving the message ($\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, ok$), sends ($\texttt{Receive}$, $\texttt{sid}_{\texttt{2AC}}$, m) to the receiver Rec.

Expire Session:

Upon receiving ($\texttt{Expire}_{\texttt{2AC}}$, $\texttt{sid}_{\texttt{2AC}}$) from some party, un-sets the variable active.

Figure 7.4: Two Anonymous Channel Functionality $\mathcal{F}_{2AC}$

Code for Two Anonymous Channel Functionality, F_{2AC},
where for $X \in \{\mathrm{Init}_1, \mathrm{Init}_2, \mathrm{Rec}\}$

Signature:

$\mathtt{sid}_{\mathtt{2AC}} = (\{\mathrm{Init}_1, \mathrm{Init}_2\}, \mathrm{Rec}, \mathtt{sid}'_{\mathtt{2AC}})$

Input:	Output:
$\mathit{receive}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_X$	$\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}$
$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_i}$	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Adv}}$
$\mathit{receive}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_X$	$\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Rec}}$
$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{Adv}}$	$\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}$

State:

$\mathit{estcond}_X \in \{\perp, \top\}$, initially $\perp$	$\mathit{okcond}_{\mathrm{Adv}} \in \{\perp, \top\}$, initially $\perp$
$\mathit{mes} \in (\{0,1\}) \cup \{\perp\}$, initially $\perp$	$\mathit{active} \in \{\perp, \top\}$, initially $\perp$
$\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$	

Tasks: $\{\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}$, $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Rec}}$,
$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Adv}}$, $\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}\}$

Figure 7.5: Code for Two Anonymous Channel Functionality $\mathcal{F}_{2AC}$, F_{2AC} (Part I)

Code for Two Anonymous Channel Functionality, $\mathrm{F}_{2\mathrm{AC}}$,
where for $X \in \{\mathrm{Init}_1, \mathrm{Init}_2, \mathrm{Rec}\}$

Transitions:

Establish Session:

```
ESS1. $receive(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_X$
        pre: $active = \bot$ and $ntask = \bot$
        eff: $estcond_X := \top$
        If $estcond_X$ for all $X$ then $active := \top$ and $ntask :=$ ESS2
        Else do nothing.
ESS2. $send(\texttt{SID}, \texttt{sid}_{\texttt{2AC}})_{\mathrm{Adv}}$
        pre: $active = \top$ and $ntask =$ ESS2
        eff: $ntask := \bot$
```

Data Sending Session:

```
DSS1. $receive(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{Init}_i} (i \in \{1, 2\})$
        pre: $active = \top$, $mes = \bot$ and $ntask = \bot$
        eff: $mes := m$ and $ntask :=$ DSS2
DSS2. $send(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, mes)_{\mathrm{Adv}}$
        pre: $okcond_{\mathrm{Adv}} = \bot$, $mes := m$ and $ntask =$ DSS2
        eff: $ntask :=$ DSS3
DSS3. $receive(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, ok)_{\mathrm{Adv}}$
        pre: $ntask =$ DSS3
        eff: $okcond_{\mathrm{Adv}} := \top$ and $ntask :=$ DSS4
DSS4. $send(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, mes)_{\mathrm{Rec}}$
        pre: $ntask =$ DSS4
        eff: $okcond_{\mathrm{Adv}}$, $mes$ and $ntask := \bot$
```

Expire Session:

```
EXS1. $receive(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_X$
        pre: $active = \top$, $mes$ and $ntask = \bot$
        eff: $ntask :=$ EXS2
EXS2. $send(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\mathrm{Adv}}$
        pre: $ntask =$ EXS2
        eff: $active, estcond_X$ and $ntask := \bot$ for all $X$
```

Figure 7.6: Code for Two Anonymous Channel Functionality $\mathcal{F}_{2\mathrm{AC}}$, $\mathrm{F}_{2\mathrm{AC}}$ (Part II)

Functionality $\mathcal{F}_{\text{DIC}}$

$\mathcal{F}_{\text{DIC}}$ proceeds as follows, running with parties $P \in \{\text{Init}, \text{Rec}\}$ and an adversary.

Establish Session:

Upon receiving $(\texttt{Establish}_{\texttt{DIC}},\ \texttt{sid}_{\texttt{DIC}})$ from some party Init, verifies that $\texttt{sid}_{\texttt{DIC}} = (\{\text{Init}, \text{Rec}\}, \texttt{sid}_{\texttt{DIC}}')$ for Rec, sends $(\texttt{SID}, \texttt{sid}_{\texttt{DIC}})$ to the adversary, and waits to receive $(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})$ from Rec. Upon receiving this message, sets a boolean variable as active.

Data Sending Session:

Upon receiving $(\texttt{Send},\ \texttt{sid}_{\texttt{DIC}},\ m)$ from $P \in \{\text{Init}, \text{Rec}\}$, and if active is set, sends $(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)$ to the adversary. Upon receiving $(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)$, sends $(\texttt{Receive},\ \texttt{sid}_{\texttt{DIC}},\ m)$ to the other party $\bar{P}$.

Expire Session:

Upon receiving $(\texttt{Expire}_{\texttt{DIC}},\ \texttt{sid}_{\texttt{DIC}})$ from either party, un-sets the variable active.

Figure 7.7: Direction-Indeterminable Channel Functionality $\mathcal{F}_{\text{DIC}}$

Code for Direction-Indeterminable Channel Functionality, $\mathrm{F}_{\mathrm{DIC}}$,
where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathtt{DIC}})$

Input:	Output:
$\mathit{receive}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_X$	$\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$
$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_X$	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$
$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{Adv}}$	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\overline{X}}$
$\mathit{receive}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_X$	$\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$

State:

$\mathit{estcond}_X \in \{\bot, \top\}$, initially $\bot$ $\quad$ $\mathit{mes} \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$
$\mathit{okcond}_{\mathrm{Adv}} \in \{\bot, \top\}$, initially $\bot$ $\quad$ $\mathit{active} \in \{\bot, \top\}$, initially $\bot$
$\mathit{ntask} \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Tasks:

$\{\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}, \mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}},$
$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\overline{X}}, \mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}\}$

Figure 7.8: Code for Direction-Indeterminable Channel Functionality $\mathcal{F}_{\mathrm{DIC}}$, $\mathrm{F}_{\mathrm{DIC}}$(Part I)

```
Code for Direction-Indeterminable Channel Functionality, F_DIC,
where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Transitions:

    Establish Session:

        ESS1. $\mathit{receive}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_X$
                pre: $\mathit{active} = \bot$ and $\mathit{ntask} = \bot$
                eff: $\mathit{estcond}_X := \top$.
                If $\mathit{estcond}_X = \top$ for all $X$ then $\mathit{active} := \top$ and $\mathit{ntask} :=$ ESS2.
                Else do nothing.
        ESS2. $\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$
                pre: $\mathit{active} = \top$ and $\mathit{ntask} =$ ESS2
                eff: $\mathit{ntask} := \bot$

    Data Sending Session:

        DSS1. $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_X$
                pre: $\mathit{active} = \top$, $\mathit{mes}$ and $\mathit{ntask} = \bot$
                eff: $\mathit{mes} := m$ and $\mathit{ntask} :=$ DSS2
        DSS2. $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$
                pre: $\mathit{okcond}_{\mathrm{Adv}} = \bot$, $m := \mathit{mes}$ and $\mathit{ntask} =$ DSS2
                eff: $\mathit{ntask} :=$ DSS3
        DSS3. $\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{Adv}}$
                pre: $\mathit{ntask} =$ DSS3
                eff: $\mathit{okcond}_{\mathrm{Adv}} := \top$ and $\mathit{ntask} :=$ DSS4
        DSS4. $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\overline{X}}$
                pre: $\mathit{ntask} =$ DSS4
                eff: $\mathit{okcond}_{\mathrm{Adv}}$, $\mathit{mes}$ and $\mathit{ntask} := \bot$

    Expire Session:

        EXS1. $\mathit{receive}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_X$
                pre: $\mathit{active} = \top$, $\mathit{mes} = \bot$ and $\mathit{ntask} = \bot$
                eff: $\mathit{ntask} :=$ EXS2
        EXS2. $\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$
                pre: $\mathit{ntask} =$ EXS2
                eff: $\mathit{active}$, $\mathit{estcond}_X$ and $\mathit{ntask} := \bot$ for all $X$
```

Figure 7.9: Code for Direction-Indeterminable Channel Functionality $\mathcal{F}_{\mathrm{DIC}}$, $\mathrm{F}_{\mathrm{DIC}}$ (Part II)

7.2 Equivalence Between DIC and 2AC

In this section, we prove that the DIC is equivalent to the 2AC under some types of schedules. To prove this, we show two reductions, DIC to 2AC and 2AC to DIC. Here, we consider the exchange of a one bit message, i.e., the message length $|m| = 1$. Informally, the reduction of DIC to 2AC is proven as described hereafter. The direction-indeterminable property is generated using two 2AC functionalities, $\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}$ and $\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}$. Here, the two senders of $\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}$ are Init and Rec, and the receiver of $\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}$ is Init. The two senders of $\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}$ are Init and Rec, and the receiver of $\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}$ is Rec. When Init sends a message to receiver Rec, Init sends the message using $\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}$ and $\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}$. More specifically, $\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}$ forwards the message to Init and $\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}$ forwards the message to Rec. The execution order of $\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}$ is selected at random by the message sender. An adversary cannot detect the direction the message was sent because Init and Rec receive the same message, m, transferred by the two 2ACs. The other reduction, 2AC to DIC, is proven as described hereafter. First, the message sending party (Init_1 or Init_2) sends message m to the other party using a DIC. Init_1 and Init_2 then send m to receiver Rec directly under some type of master schedule. An adversary cannot detect which is the sender because the direction the message was sent between senders Init_1 and Init_2 is indeterminable.

7.2.1 Reduction of DIC to 2AC

Let π_{DIC} be a protocol of the DIC. We assume that $M_{\pi_{\mathrm{DIC}}}$, the master schedule of π_{DIC}, is any schedule. Let $\mathrm{Init}_{\mathrm{DIC}}$ and $\mathrm{Rec}_{\mathrm{DIC}}$ be the initiator code and receiver code for a real system, see Fig.7.10, Fig.7.11, and Fig.7.12, and Fig.7.13 and Fig.7.14, respectively. Let $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$ be the initiator code and receiver code for an ideal system, see Fig.7.17 and Fig.7.18, and Fig.7.19 and Fig.7.20, respectively. Finally, let $\mathrm{Adv}_{\mathrm{DIC}}$ and $\mathrm{Sim}_{\mathrm{DIC}}$ be the adversary code and the simulator code in Fig.7.15 and Fig.7.16, and Fig.7.21, and Fig.7.22, respectively. Let $\mathrm{Real}_{\mathrm{DIC}}$ and $\mathrm{Ideal}_{\mathrm{DIC}}$ be a DIC protocol system and a DIC functionality system, respectively. These are defined below.

$$\mathrm{Real}_{\mathrm{DIC}} := \mathit{hide}(\mathrm{Init}_{\mathrm{DIC}} \| \mathrm{Rec}_{\mathrm{DIC}} \| \mathrm{Adv}_{\mathrm{DIC}} \| \mathrm{F}^{\mathrm{I}}_{\mathrm{2AC}} \| \mathrm{F}^{\mathrm{R}}_{\mathrm{2AC}}, \{\mathrm{rand}(*)\}),$$
$$\mathrm{Ideal}_{\mathrm{DIC}} := \overline{\mathrm{Init}_{\mathrm{DIC}}} \| \overline{\mathrm{Rec}_{\mathrm{DIC}}} \| \mathrm{Sim}_{\mathrm{DIC}} \| \mathrm{F}_{\mathrm{DIC}}.$$

Tasks $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$ relay the input messages from the environment to the ideal functionality task and relay the received messages from the ideal functionality task to the environment as interface parties in the ideal system.

Theorem 4. *DIC protocol system* $\mathrm{Real}_{\mathrm{DIC}}$ *perfectly hybrid-implements DIC functionality system* $\mathrm{Ideal}_{\mathrm{DIC}}$ *with respect to an adaptive adversary under any master schedule (DIC is reducible to 2AC with respect to an adaptive adversary under any master schedule).*

Let ϵ_{R} and ϵ_{I} be discrete probability measures on finite executions of $\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}$ and $\mathrm{Ideal}_{\mathrm{DIC}} \| \mathrm{Env}$, respectively. We prove Theorem 4 by showing that ϵ_{R} and ϵ_{I} satisfy the trace distribution property, $\mathit{tdist}(\epsilon_{\mathrm{R}}) = \mathit{tdist}(\epsilon_{\mathrm{I}})$. Here, we define correspondence relation R between the states in $\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}$ and the states in $\mathrm{Ideal}_{\mathrm{DIC}} \| \mathrm{Env}$. We say $(\epsilon_{\mathrm{R}}, \epsilon_{\mathrm{I}}) \in R$ if and only if for every $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$, all of the state correspondences in Tables 7.1, 7.2 and 7.3 hold. We then prove R is a simulation relation in Lemma 1.

Lemma 1. *Relation R defined above is a simulation relation from* $\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}$ *to* $\mathrm{Ideal}_{\mathrm{DIC}} \| \mathrm{Env}$ *under master schedule* $M_{\pi_{\mathrm{DIC}}}$.

Proof. We prove that R is a simulation relation from $\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}$ to $\mathrm{Ideal}_{\mathrm{DIC}} \| \mathrm{Env}$ using the mapping corrtasks, $R^{*}_{\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}} \times R_{\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}} \rightarrow R^{*}_{\mathrm{Ideal}_{\mathrm{DIC}} \| \mathrm{Env}}$, which is defined below (hereafter $T =_{\mathrm{corr.}} T'$ is used as an alternative way to write corrtask $(\rho, T) = T'$).

The task sequence of system $\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}$ perfectly corresponds to that of system $\mathrm{Ideal}_{\mathrm{DIC}} \| \mathrm{Env}$ under schedule $M_{\pi_{\mathrm{DIC}}}$. Formally, to prove that R is a simulation relation from $\mathrm{Real}_{\mathrm{DIC}} \| \mathrm{Env}$ to $\mathrm{Ideal}_{\mathrm{DIC}} \| \mathrm{Env}$, we must show that R satisfies the start condition and step condition.

- **Start condition**
  It is true that the respective start states of s and u in $\text{Real}_{\text{DIC}}||\text{Env}$ and $\text{Ideal}_{\text{DIC}}||\text{Env}$ are on the Dirac measures. That is, the start states of s and u satisfy relation R because the start states of s and u are all $\perp$ for each task on master schedule $M_{\pi_{\text{DIC}}}$. Therefore, the trace distribution property holds.

- **Step condition**
  If $(\epsilon_{\text{R}}, \epsilon_{\text{I}}) \in R$, $\rho \in R^*_{\text{Real}_{\text{DIC}}||\text{Env}}$, ϵ_{R} is consistent with ρ, ϵ_{I} is consistent with $full(corrtasks)(\rho)$, and $T \in \text{Real}_{\text{DIC}}||\text{Env}$. Then there exist the following.

  - Probability measure p on countable index set I,
  - Probability measures $\epsilon'_{\text{R},j}$, $j \in I$, on finite executions of $\text{Real}_{\text{DIC}}||\text{Env}$, and
  - Probability measures $\epsilon'_{\text{I},j}$, $j \in I$, on finite executions of $\text{Ideal}_{\text{DIC}}||\text{Env}$,

  such that

  - For each $j \in I$, $\epsilon'_{\text{R},j} \; R \; \epsilon'_{\text{I},j}$,
  - $\Sigma_{j\in I} p(j)(\epsilon'_{\text{R},j}) = apply(\epsilon_{\text{R}}, T)$, and
  - $\Sigma_{j\in I} p(j)(\epsilon'_{\text{I},j}) = apply(\epsilon_{\text{I}}, corrtask(\rho, T))$.

Task Correspondence

For any $(\rho, T) \in (R^*_{\text{Real}_{\text{DIC}}||\text{Env}} \times R_{\text{Real}_{\text{DIC}}||\text{Env}})$, the following task correspondences, which are also summarized in Table 7.4, hold.

1. **Establish Session**

   (a) $\text{Init}_{\text{DIC}}.send(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}^{\texttt{I}}_{\texttt{2AC}})_{\text{F}^{\text{I}}_{\text{2AC}}}$
   $\cdot\, \text{Init}_{\text{DIC}}.send(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}^{\texttt{R}}_{\texttt{2AC}})_{\text{F}^{\text{R}}_{\text{2AC}}}$
   $=_{\text{corr.}} \overline{\text{Init}_{\text{DIC}}}.send(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$

Let $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ be $send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}}$ where $X \in \{\mathtt{I}, \mathtt{R}\}$ and $send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$, respectively. We assume that for each start state in $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. The preconditions for $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are the same as $ntask = \mathrm{ESS2}$. $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is enabled (or disabled) in s (resp., u) if and only if $s.\mathrm{Init}_{\mathrm{DIC}}.ntask = \mathrm{ESS2}$ (resp. $u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.ntask = \mathrm{ESS2}$). From ($i$) and ($j$) in Table 7.1, the state correspondences imply that $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ for $X \in \{\mathtt{I}, \mathtt{R}\}$. Note that in the establish session, the order of $\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}$ is fixed with $\mathrm{F}^{\mathtt{I}}_{2\mathrm{AC}}$ and $\mathrm{F}^{\mathtt{R}}_{2\mathrm{AC}}$ in this order.

i. Disable Case:
   Let I and p be the set that has a single element and Dirac measure on I, respectively. Let $\epsilon'_{\mathrm{R},1} = \epsilon'_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon'_{\mathrm{I}}$. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, we obtain $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ from relation $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. The trace distribution equivalence property, $tdist(\epsilon'_{\mathrm{R}}) = tdist(\epsilon'_{\mathrm{I}})$, also holds since $tdist(\epsilon_{\mathrm{R}}) = tdist(\epsilon_{\mathrm{I}})$ under $M_{\pi_{\mathrm{DIC}}}$.

ii. Enable Case:
   Let q denote the state of precondition $ntask = \mathrm{ESS2}$. Let $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ be actions enabled.
   Let I and p be a set that has a single element and the Dirac measure on I, respectively. Let $\epsilon'_{\mathrm{R},1} = \epsilon'_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon'_{\mathrm{I}}$. Here, we establish the property of R for ϵ'_{R} and ϵ'_{I} to show that $(\epsilon'_{\mathrm{R}}, \epsilon'_{\mathrm{I}}) \in R$. To establish the property, consider any state $s' \in \mathrm{supp.lst}(\epsilon'_{\mathrm{R}})$ and $u' \in \mathrm{supp.lst}(\epsilon'_{\mathrm{I}})$. Let s be any state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ such that $s' \in \mathrm{supp}(\mu_{\mathrm{s}})$ where $(s, \zeta, \mu_s) \in \mathrm{Real}_{\mathrm{DIC}}||\mathrm{Env}$. Let u be any state in $\mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ such that $u' \in \mathrm{supp}(\mu_{\mathrm{u}})$ where $(u, corrtask(\rho, \zeta), \mu_u) \in \mathrm{Ideal}_{\mathrm{DIC}}||\mathrm{Env}$.
   It is true that $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ updates $\mathrm{Init}.active$ to $\top$ and $\mathrm{Init}_{\mathrm{DIC}}.ntask$ to $\perp$ from the definition of the effect of $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ for $X \in \{\mathrm{Init}, \mathrm{Rec}\}$. Similarly, $\mathtt{T}_{\mathtt{IDEAL}}$ updates $\overline{\mathrm{Init}_{\mathrm{DIC}}}.active$ to $\top$

and $\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{ntask}$ to $\perp$ from the definition of the effect of $\mathtt{T}_{\mathtt{IDEAL}}$. From the state equivalences of (i) and (j) in Table 7.1, we have $u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{active} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{active}$ and $u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{ntask} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{ntask}$. By the definitions of $\mathrm{Init}_{\mathrm{DIC}}$ and $\overline{\mathrm{Init}_{\mathrm{DIC}}}$, $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is a unique action that updates the state of *active* of $\mathrm{Real}_{\mathrm{DIC}}$ (resp., $\mathrm{Ideal}_{\mathrm{DIC}}$). We then obtain that $u'.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{active} = s'.\mathrm{Init}_{\mathrm{DIC}}.\mathit{active}$ and $u'.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{ntask} = s'.\mathrm{Init}_{\mathrm{DIC}}.\mathit{ntask}$. Therefore, we obtain the trace distribution property $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

(b) $\mathrm{Rec}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathrm{I}}_{\mathrm{2AC}}}$
$\cdot\, \mathrm{Rec}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathrm{R}}_{\mathrm{2AC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
This is analogous to the case of 1a. The states of precondition and effect for both expressions are the same. More specifically, the precondition and effect of the real task are the same as those for the ideal task based on pre:(n), and eff:(n) and (m) in Table 7.1. So, these tasks correspond (Hereafter, the descriptions of precondition and effect are referred to as pre: and eff:, respectively).

   i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold based on each task definition and the state correspondence pre:(n). Therefore, we obtain $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

   ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence pre:(n), and state correspondences eff:(m) and (n), we have $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

(c) $\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}})_{\mathrm{Adv}} \cdot \mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}})_{\mathrm{Adv}}$ $=_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$

The precondition and effect of the real tasks are identical to those for the ideal tasks. The preconditions for each of the tasks on the left side of the equation are $\mathit{active} = \top$ and $\mathit{ntask} = \mathrm{ESS2}$, respectively. The task for the expression on the right side of the equation also has the same preconditions. The effects of the tasks on the left side of the equation are $\mathit{ntask} := \bot$. This effect is also the same for the task on the right.

Let $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ be $\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}$ for $X \in \{\mathrm{Init}, \mathrm{Rec}\}$. Let $\mathtt{T}_{\mathtt{IDEAL}}$ be $\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$. We show that $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. We consider that for each state in $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. Then, $\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}$ is enabled (or disabled) in s if and only if $s.\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}.\mathit{active} = \top$ and $s.\mathtt{T}^{\mathtt{X}}_{\mathtt{REAL}}.\mathit{ntask} = \mathrm{ESS2}$. The pre:($d$) and ($f$) in Table 7.1 imply that $\mathtt{T}_{\mathtt{IDEAL}}$ is uniformly enabled or disabled. The rest of this proof is similar to the case of 1a.

2. **Data Sending Session**

Here, we consider the case that Env sends the data sending message in $\mathrm{Init}_{\mathrm{DIC}}$. The case that Env sends a data sending message in $\mathrm{Init}_{\mathrm{DIC}}$ is analogous to the case for $\mathrm{Rec}_{\mathrm{DIC}}$. The task sequence in each world is shown in Table 7.5 and Table 7.6. The task sequence of the Real Execution corresponds to that of the Ideal Execution. Note that the order of the $\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}$ for sending the message is not fixed so that the message direction cannot be distinguished. To fix the order of $\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}$, we use $\mathrm{F}_{\mathrm{SRC}}$.

The flow of the states in each task is shown in Tables 7.7 and 7.8 for each world. From the initial values and final values in Tables 7.7 and 7.8, we obtain the result of the state equivalence in 7.1. That is, if the state equivalences in 7.1 hold before the task sequence is enabled (or disabled), the state equivalences in 7.1 after the task sequence is completed also hold.

(a) Disable Case: This is a trivial case because all the states of the parties are $\perp$. The states do not change before or after the protocol starts in each world. That is, Env inputs no message to the parties. Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from the task definition and the state correspondence. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(b) Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definition and the follow flow of states in Tables 7.7 and 7.8, it is oblivious that the initial state is the same as the final state for each task of each world. In addition, the states of the real task are also the same as the states of the ideal world after the data sending session is executed. That is, $(a) \sim (n)$ in Table 7.1 hold. Therefore, we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

3. **Expire Session**

(a) $\mathrm{Init}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}}$
$\cdot\, \mathrm{Init}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$
$=_{\mathrm{corr.}}\ \overline{\mathrm{Init}_{\mathrm{DIC}}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
The states of precondition and effect for the task on the left, $send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$, are the same as those for the task on the right, $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$, where $ntask = \mathrm{EXS2}$. That is, if (i) and (j) in Table 7.1 hold, then these tasks are enabled (or disabled) in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

Note that the order of the $\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}$ is fixed with $\mathrm{F}^{\mathtt{I}}_{2\mathrm{AC}}$ and $\mathrm{F}^{\mathtt{R}}_{2\mathrm{AC}}$ in this order.

The precondition of both tasks is $ntask = \mathrm{EXS2}$. More specifically, the precondition and effect of the real tasks are the same as the ideal task from pre:(j), and eff:(i) and (j), respectively.

i. Disable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence pre: (j). Therefore, we obtain $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

ii. Enable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence pre: (j), and the state correspondences eff: (i) and (j), we have that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

(b) $\mathrm{Rec}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{I}}_{2\mathrm{AC}}}$
$\cdot \mathrm{Rec}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{R}}_{2\mathrm{AC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

This is similar to the case of 3a. The precondition states of both expressions are the same as $ntask = \mathrm{EXS2}$. More specifically, the precondition and effect of the real task are the same as those for the ideal task based on pre: (n), and eff:(m) and (n), respectively. So, these tasks correspond. Note that the order of $\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}$ is fixed with $\mathrm{F}^{\mathtt{I}}_{2\mathrm{AC}}$ and $\mathrm{F}^{\mathtt{R}}_{2\mathrm{AC}}$ in this order.

i. Disable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence pre: (n). Therefore, we obtain

$trace(\epsilon'_R) = trace(\epsilon'_I)$.

ii. Enable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence pre: (n), and the state correspondences eff: (m) and (n), we have that $\epsilon'_R = \epsilon_R$ and $\epsilon'_I = \epsilon_I$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_R) = trace(\epsilon'_I)$.

(c) $F^{\mathtt{I}}_{2AC}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}})_{\mathrm{Adv}}$
$\cdot F^{\mathtt{R}}_{2AC}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}})_{\mathrm{Adv}}$
$=_{\mathrm{corr.}} F_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$

The precondition and effect of the real task are the same as those for the ideal task. The precondition is only $ntask = \mathrm{EXS2}$ and the effect is $active := \perp$, $estcond_X := \perp$ for all X (and $estcond_{\mathrm{Init}_i}, estcond_{\mathrm{Rec}} := \perp$ for all i in $F^{\mathtt{X}}_{2AC}$) and $ntask := \perp$. From (f) in Table 7.1, these tasks are enabled (or disabled) in every state in supp.lst$(\epsilon_R) \cup$ supp.lst(ϵ_I). The rest of this proof is analogous to the case of 3a.

i. Disable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_R = \epsilon_R$ and $\epsilon'_I = \epsilon_I$. Here, the start and step conditions of the simulation relation R hold from each task definition and the state correspondence pre: (f). Therefore, we obtain $trace(\epsilon'_R) = trace(\epsilon'_I)$.

ii. Enable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence pre: (f), and the state correspondences eff: (a), (b), (d), and (f), we have that $\epsilon'_R = \epsilon_R$ and $\epsilon'_I = \epsilon_I$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_R) = trace(\epsilon'_I)$.

Environment Env

From the task definitions and state correspondence (o) in Table 7.1, the provability measures of both tasks are uniformly enabled or disabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

> **Claim 1** The state of Env remains static in all states in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}). Let q_e denote this state of Env. This follows from state correspondence (o).
>
> **Claim 2** If T is a task of Env, then T is either enabled or disabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}) (simultaneously). Furthermore, if T is enabled in all states in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}), then:
>
> 1. There exists unique action $a \in$ T that is enabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).
> 2. There exists a unique transition of Env from q_e with action a. Let $tr_e = (q_e, a, \mu_e)$ be this transition.

By considering Claim 7.2.1, task T of Env is uniformly enabled or disabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}). If T is disabled, let $I = 1$, we obtain $\epsilon'_{\mathrm{R},1} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon_{\mathrm{I}}$, and the result is $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ since we have $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. If T is enabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}), Claim 7.2.1 implies that there exists unique action a in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}) and transition tr_e of Env from q_e enabled with action a where $tr_e = (q_e, a, \mu_e)$.

Non Corrupted Case:

1. a is an input / output action of Init. We assume that a is an input action such as $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$, $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ and $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$, and an output action such as $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Init}}$.

   Let s be any state such that $s' \in$ supp(μ_{s}) where $(s, a, \mu_s) \in D_{\mathrm{Real_{DIC}}||\mathrm{Env}}$. Let u be any state such that $u' \in$ supp(μ_{u}) where $(u, a, \mu_u) \in$

$D_{\mathrm{Ideal}_{\mathrm{DIC}}||\mathrm{Env}}$. For each a, we check that the state correspondence for sf and u' holds if that for s and u holds. If each a is input from Env, then the precondition and effect are exactly the same between the real task and ideal task. For example, if the input message is $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$, then the precondition is $active, ntask = \perp$ and the effect is $ntask :=$ ESS2. The real task state correspond to those for the ideal task. So, in the case of the enabled (or disabled), it is hold that the state correspondences of (o), (i) and (j) for s' and u', if those for s and u hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$. This result also works well in the case of $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$, $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$ and $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Init}}$.

2. a is an input / output action of Rec. We assume that a is an input action such as $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$, $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$, $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$, and $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Rec}}$. This is analogous to the case of 1.

3. a is an input / output action of Adv. This means that $a = input(g)_{\mathrm{Adv}}$ for some fixed g. For example, g is a corrupt message for some $party \in \{\mathrm{Init}, \mathrm{Rec}\}$. From the fact that the state correspondences $(A) \sim (T)$ for s and u holds, we obtain that the state corresponces for s' and u' holds. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

4. a is an internal or an output action of Env. Task a in the real world is identical to that in the ideal world. From the fact that state correspondence (o) for s and u holds, we obtain that state correspondence (o) for s' and u' holds. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R},j}) = trace(\epsilon'_{\mathrm{I},j})$.

Corrupted Case:

1. a is an input action of Adv and $party \in \{\mathrm{Init}, \mathrm{Rec}\}$ Here, the party is included in the case of $\mathrm{Init} \wedge \mathrm{Rec}$. Let q_{Adv} be the state of Adv

or Sim, which is the same in all supp.lst(ϵ_R) ∪ supp.lst(ϵ_I). Let $tr_{\text{Adv}} = (q_{\text{Adv}}, a, \mu_{\text{Adv}})$ be a transition of Adv with action a from q_{Adv}. From Claim 7.2.1, tr_{Adv} is a unique transition. Here, we suppose that supp(($\mu_e \times \mu_{\text{Adv}}$)) is the pair set $\{(q_{1,j}, q_{2,j}) : j \in I\}$, where I is a countable set. Let p be the probability measures such that for each j, $p(j) = (\mu_e \times \mu_{\text{Adv}})(q_{1,j}, q_{2,j})$. For each j, let $\epsilon'_{R,j}$ be $\epsilon'_{1,j}(\alpha) = \epsilon_1(\alpha')$, where $\alpha \in \text{supp}(\epsilon'_1)$ such that `lst`(α).Env $= q_{1,j}$ and `lst`(α).Adv $= q_{2,j}$. The $\epsilon'_{2,j}$ is analogously constructed from ϵ'_2.

The rest of this proof is the same as that for case 1 by considering the state correspondence in each case $party \in \{\text{Init}, \text{Rec}, \text{Init} \wedge \text{Rec}\}$. Finally, we obtain the trace distribution property, $trace(\epsilon'_{R,j}) = trace(\epsilon'_{I,j})$.

Adversary Adv

From the task definitions and the state correspondences, $(A) \sim (T)$, in Table 7.2, the provability measures of both tasks are uniformly enabled or disabled in every state in supp.lst(ϵ_R) ∪ supp.lst(ϵ_I).

Claim 3 The state of Adv or Sim is the same in all states in supp.lst(ϵ_R) ∪ supp.lst(ϵ_I). Let q_{Adv} denote this state of Adv and Sim. This follows from state correspondence of Sim.

Claim 4 If T is a task of Adv, then T is either enabled or disabled in every state in supp.lst(ϵ_R) ∪ supp.lst(ϵ_I). Furthermore, if T is enabled in all states in supp.lst(ϵ_R) ∪ supp.lst(ϵ_I), then:

1. There is unique action $a \in$ T that is enabled in every state in supp.lst(ϵ_R) ∪ supp.lst(ϵ_I).
2. There is a unique transition of Adv from q_{Adv} with action a and let $tr_{\text{Adv}} = (q_{\text{Adv}}, a, \mu_{\text{Adv}})$ be this transition.

By considering Claim.7.2.1, task T of Adv is uniformly enabled or disabled in every state in supp.lst(ϵ_R) ∪ supp.lst(ϵ_I). If T is disabled, let

$I = 1$, we obtain $\epsilon'_{\mathrm{R},1} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon_{\mathrm{I}}$, and the result is $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ since we have $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. If T is enabled, T is enabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}). Claim 7.2.1 implies that there is unique action a in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}) and transition tr of Adv from q_e enabled with action a where $tr_{\mathrm{Adv}} = (q_{\mathrm{Adv}}, a, \mu_{\mathrm{Adv}})$. The following cases for the "Non Corrupted Case" and "Corrupted Case" can be considered.

Non Corrupted Case:

1. a is an input action of Env. From the fact that the state correspondences, $(A) \sim (T)$, for s and u holds, we obtain that the state correspondences for s' and u' holds. Therefore, we obtain the trace distribution property: $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

2. a is an input or output action of functionality. This case concerns the messages $receive(\mathtt{SID}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$, $receive(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$, and $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$. The rest of this proof is analogous to the case of 1. From the fact that the state correspondences, $(A) \sim (T)$, for s and u holds, we obtain that the state correspondences for s' and u' holds. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

3. a is either an output action of Adv that is not an input action of Env, Init, Rec, or functionality task, or is an internal action of Adv. This case concerns "new" tasks. The rest of this proof is analogous to the case of 1. From the fact that the state correspondences, $(A) \sim (T)$, for s and u holds, we obtain that the state correspondences for s' and u' holds. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

4. a is an output action of $out(*)_{adv}$. This case also works well although this action may affect Env. However, the transition of Env $tr_e = (q_e, a, \mu_e)$ is unique from Claim 7.2.1. Claim 7.2.1 also says that the state of Env remains static in all states in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

This follows from state correspondence o. Similarly, from the definition and some claims, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Corrupted Case:

This is the case that the static and adaptive adversary Adv corrupts $party \in \{\mathrm{Init}, \mathrm{Rec}\}$.

1. a is input or output action $in(*)_{party}$ or $out(*)_{party}$ of corrupted party, $party \in \{\mathrm{Init}, \mathrm{Rec}\}$, respectively. This case also works well from the Claim 7.2.1 and state correspondence in Table 7.1 ∼ 7.3.

Perfect Simulation

Another task of $\mathrm{Sim}_{\mathrm{DIC}}$ is the $simulation(*)$ task. By using $simulation(*)$ effectively, the simulation of $\mathrm{Sim}_{\mathrm{DIC}}$ perfectly mimics the establish session, the data sending session, and the expire session with respect to no corruption, static corruption and adaptive corruption by an adversary.

1. **No Corruption**

   (a) **Establish Session** First, in the establish session, environment Env sends establish message $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Init}}}$ and message $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$ to initiator $\overline{\mathrm{Init}}_{\mathrm{DIC}}$ and receiver $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$, respectively. They send establish session messages $send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. The functionality sends $send(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ to simulator $\mathrm{Sim}_{\mathrm{DIC}}$. After $\mathrm{Sim}_{\mathrm{DIC}}$ receives $send(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ starts $simulation(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})$ in his simulation world under the following policy.

   Simulation Policy

   i. After receiving $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes the following simulation.

A. prepares dummy parties Init, Rec, and Adv and ideal functionality task $\mathrm{F}_{2\mathrm{AC}}$.

B. $\mathrm{Sim}_{\mathrm{DIC}}$ inputs messages $in(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}}$ and $in(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Rec}}$ to Init and Rec, respectively.

C. $\mathrm{Sim}_{\mathrm{DIC}}$ makes Init (resp., Rec) send message $send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}}$ to $\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}$ for each $X \in \{\mathrm{I}, \mathrm{R}\}$ in this order(I then R), respectively.

D. $\mathrm{Sim}_{\mathrm{DIC}}$ makes $\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}$ send $send(\mathtt{SID}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{Adv}}$ to Adv.

Task Correspondence of Simulation

i. $\mathrm{Init}_{\mathrm{DIC}}.send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}}$

pre: $ntask = \mathrm{ESS2}$; (L);

eff: $active := \top$ and $ntask := \bot$; $(M), (L)$;

ii. $\mathrm{Rec}_{\mathrm{DIC}}.send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}}$

pre: $ntask = \mathrm{ESS2}$; (Q);

eff: $active = \top$ and $ntask := \bot$; $(P), (Q)$;

iii. $\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}.send(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{2\mathrm{AC}}.send(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}_{\mathrm{DIC}}}$

pre: $ntask = \mathrm{ESS2}$; (I);

eff: $active$, $estcond_X$ and $ntask := \bot$ for all X ; $(D) \sim (G)$, (I);

The simulation of the establish session is perfectly executed by $simulation(*)$ of $\mathrm{Sim}_{\mathrm{DIC}}$. Finally, the parties establish two 2ACs in the simulation world.

(b) **Data Sending Session** Next, in the data sending session, Env sends message $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Init}}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}}}$) to $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ (or $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$). $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ sends $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$

to F_{DIC}. F_{DIC} then sends $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$ to $\mathrm{Sim}_{\mathrm{DIC}}$. After $\mathrm{Sim}_{\mathrm{DIC}}$ receives $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes task $\mathit{simulation}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)$ in his simulation world under the following policy.

Simulation Policy

i. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{F_{\mathrm{DIC}}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes the following simulation.

   A. $\mathrm{Sim}_{\mathrm{DIC}}$ executes $\mathit{random}(*)$ and selects message input party $\mathit{party} \in \{\mathrm{Init}, \mathrm{Rec}\}$ (The following discussion assumes $\mathit{party} = \mathrm{Init}$. The case of $\mathit{party} = \mathrm{Rec}$ is analogous).

   B. $\mathrm{Sim}_{\mathrm{DIC}}$ inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}}$ to Init.

   C. $\mathrm{Sim}_{\mathrm{DIC}}$ makes Init generate random value $s \in \{0, 1\}$.

      - If $s = 0$, $\mathrm{Sim}_{\mathrm{DIC}}$ makes Init send messages $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}}, m)_{F^{\mathtt{I}}_{\mathrm{2AC}}}$ and $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{F^{\mathtt{R}}_{\mathrm{2AC}}}$ to $F^{\mathtt{I}}_{\mathrm{2AC}}$ and $F^{\mathtt{R}}_{\mathrm{2AC}}$ in this order, respectively.
      - Else ($s = 1$), $\mathrm{Sim}_{\mathrm{DIC}}$ makes Init send message $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{F^{\mathtt{R}}_{\mathrm{2AC}}}$ and $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}}, m)_{F^{\mathtt{I}}_{\mathrm{2AC}}}$ to $F^{\mathtt{R}}_{\mathrm{2AC}}$ and $F^{\mathtt{I}}_{\mathrm{2AC}}$ in this order, respectively.

   D. $\mathrm{Sim}_{\mathrm{DIC}}$ makes $F^{\mathtt{X}}_{\mathrm{2AC}}$ receive $\mathit{receive}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Init}}$ and makes $F^{\mathtt{X}}_{\mathrm{2AC}}$ send $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Adv}}$.

   E. If $\mathit{send}(\mathtt{Response}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, \mathit{ok})_{F^{\mathtt{X}}_{\mathrm{2AC}}}$ to $F^{\mathtt{X}}_{\mathrm{2AC}}$ is received from Adv, $\mathrm{Sim}_{\mathrm{DIC}}$ continues the following.

   F. $\mathrm{Sim}_{\mathrm{DIC}}$ makes $F^{\mathtt{X}}_{\mathrm{2AC}}$ receive $\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{Adv}}$ and makes $F^{\mathtt{X}}_{\mathrm{2AC}}$ send $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Init}}$ and $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Rec}}$ to Init and Rec, respectively.

   G. $\mathrm{Sim}_{\mathrm{DIC}}$ makes Init and Rec receive $\mathit{receive}(\mathtt{Receive}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{F^{\mathtt{I}}_{\mathrm{2AC}}}$ and $\mathit{receive}(\mathtt{Receive}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{F^{\mathtt{R}}_{\mathrm{2AC}}}$, respectively.

   H. $\mathrm{Sim}_{\mathrm{DIC}}$ makes Rec output $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, r)_{\mathrm{Rec}}$.

ii. $\mathrm{Sim}_{\mathrm{DIC}}$ executes $\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{F_{SC}}}$.

Task Correspondence of Simulation

Init i. $(\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$; pre: $\mathit{active} = \top$, smes and $\mathit{ntask} = \bot$:(J),(L),(M) ; eff:$\mathit{smes} := m$, $\mathit{ntask} :=$ DSS2:(J), (M))

ii. ($\mathit{rand}(*)$; pre: ; eff: $\mathit{ntask} := \bot$:(M)) (This task is used to select the order of the next task $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$. If $\mathit{rand}(*)$ outputs 0 then Init executes $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}}$ and $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$ in this order. Else, Init executes $\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$ first).

iii. ($\mathit{send}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$; pre: $m := \mathit{smes}$ and $\mathit{ntask} =$ DSS2: (M); eff: $\mathit{ntask} := \bot$:(M))

$\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}$ i. ($\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}}$; pre:$\mathit{active} = \top$, $\mathit{mes} = \bot$ and $\mathit{ntask} = \bot$: (G), (H), (I); eff:$\mathit{mes} := m$ and $\mathit{ntask} :=$ DSS2:(H),(I))

ii. ($\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Adv}}$; pre:$\mathit{okcond}_{\mathrm{Adv}} = \bot$, $\mathit{mes} := m$ and $\mathit{ntask} =$ DSS2:(F), (H), (I); eff:$\mathit{ntask} :=$ DSS3: (I))

Adv i. ($\mathit{receive}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$; pre: $\mathit{active} = \top$, $\mathit{ntask} = \bot$: (R), (S); eff:$\mathit{smes}_X := \mathit{mes}$ and $\mathit{ntask} :=$ DSS2: (S),(T))

ii. ($\mathit{send}(\mathtt{Response}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$; pre: $\mathit{ntask} =$ DSS2: (S) ; eff: $\mathit{ntask} := \bot$: (S))

$\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}$ i. ($\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{Adv}}$, pre:$\mathit{ntask} =$ DSS3:(I); eff:$\mathit{okcond}_{\mathrm{Adv}} := \top$ and $\mathit{ntask} :=$ DSS4:(F),(I))

ii. ($\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Rec}}$; pre:$\mathit{ntask} =$ DSS4:(I); eff:$\mathit{okcond}_{\mathrm{Adv}}$, mes and $\mathit{ntask} := \bot$:(F),(H),(I))

Init i. ($\mathit{receive}(\mathtt{Receive}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}}$; pre: $\mathit{active} = \top$, rmes and $\mathit{ntask} = \bot$:(K),(L),(M) ;eff:$\mathit{smes} := \bot$ and $\mathit{ntask} := \bot$:(J),(M))

Rec i. ($receive(\mathtt{Receive}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$; pre: $active = \top$, $rmes$ and $ntask = \bot$:(O),(P),(Q) ;eff:$rmes := m$ and $ntask :=$ DSS4 :(O),(Q))

ii. ($out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Rec}}$; pre:$r := rmes$ and $ntask =$ DSS4 :(Q) ;eff:$rmes$ and $ntask := \bot$:(O),(Q))

Simulator $\mathrm{Sim}_{\mathrm{DIC}}$ executes the above-mentioned process to mimic the real world under the simulation policy. The state correspondences in Table. 7.2 and 7.3 work well. The key point of this simulation is as follows. To mimic the real world, the simulator activates the parties that execute the tasks of the real world. Moreover, in order not to distinguish the output trace, the simulator simulates the real world in his simulation world by using task codes. In the real world, Init uses two 2ACs without allowing an adversary to identify the direction in which the message was sent. In the simulation world, $\mathrm{Sim}_{\mathrm{DIC}}$ can mimic the output that $\mathrm{Adv}_{\mathrm{DIC}}$ outputs in the real world. That is, the trace distributions of each world, the real world and the ideal world, are indistinguishable. In other words, since each task correspondence and the state correspondence work well, the following property works well: $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(c) **Expire Session** Finally, in the expire session, Env sends message $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Init}}}$ and $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$ to $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$, respectively. They relay message $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. After receiving $receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ from $\mathrm{F}_{\mathrm{DIC}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes task $simulation(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})$ in his simulation world under the following policy.

Simulation Policy

i. After receiving $receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes the following simulation.

A. $\mathrm{Sim}_{\mathrm{DIC}}$ inputs messages $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$ and $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$ to Init and Rec, respectively.

B. $\mathrm{Sim}_{\mathrm{DIC}}$ makes Init (resp., Rec) send $\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}}$ and $\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$ to $\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}$ and $\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}$ in this order, respectively.

C. $\mathrm{Sim}_{\mathrm{DIC}}$ makes $\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}$ send $\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ to Adv.

Task Correspondence of Simulation

i. $\mathrm{Init}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
pre: $\mathit{ntask} = \mathrm{EXS2}$; (M);
eff: active and $\mathit{ntask} := \perp$; $(L), (M)$;

ii. $\mathrm{Rec}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
pre: $\mathit{ntask} = \mathrm{EXS2}$; (Q);
eff: active and $\mathit{ntask} := \perp$; $(R), (Q)$;

iii. $\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}_{\mathrm{DIC}}}$
pre: $\mathit{ntask} = \mathrm{EXS2}$; (I);
eff: active, $\mathit{estcond}_X$ and $\mathit{ntask} := \perp$ for all X ; $(D) \sim (G)$, (I);

We assume that the state correspondences in Table 7.2 and 7.3 hold. From 3a, 3b and 3c, the state correspondences also holds after the simulation by $\mathrm{Sim}_{\mathrm{DIC}}$. That is, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

2. **Static Corruption** This type of corruption is divided into the following three cases: only Init is corrupted by Adv, only Rec is corrupted by Adv and both parties are corrupted by Adv. Note that these cases occur before the protocol starts. The adversary does not corrupt any of the parties once Env starts the protocol.

   (a) **Only** Init **is corrupted by** Adv. This case means that $\mathrm{Adv}_{\mathrm{DIC}}$ corrupts only Init before the protocol starts. So, the remaining steps are identical to those for the above-mentioned No Corruption Case without the message input party selection.

i. After receiving the corrupt message from Env, $\mathrm{Sim}_{\mathrm{DIC}}$ prepares a situation in which only Init is corrupted and adds the following policy before 1(a)iB. $\mathrm{Sim}_{\mathrm{DIC}}$ makes Adv corrupt Init.

ii. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\mathit{party} \in \{\mathrm{Init}, \mathrm{Rec}\}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes the following simulation.

   A. If the message is input to corrupted party Init, $\mathrm{Sim}_{\mathrm{DIC}}$ inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init in his simulation world.

   B. Else the message is input to Rec, and $\mathrm{Sim}_{\mathrm{DIC}}$ inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec.

   C. The remaining steps are the same as those in the simulation of the No Corrupted Case.

iii. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\overline{\mathrm{Init}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Init}}}$.

In this case, the $\mathrm{Adv}_{\mathrm{DIC}}$ and $\mathrm{Sim}_{\mathrm{DIC}}$ can identify the direction that the message is sent from Init to Rec or from Rec to Init. However, the simulation is perfectly executed. If the protocol executed the establish session, data sending session, and expire session, in any case, the simulator can simulate the real world and the movement of Adv. That is, the simulation is perfectly executed by $\mathrm{Sim}_{\mathrm{DIC}}$. From the Task Correspondence in 7.2.1, the state correspondences in 7.1, 7.2, and 7.3 hold in this case. That is, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$ holds.

(b) **Only** Rec **is corrupted by** Adv**.** This case is analogous to the case of 2a. This case means that $\mathrm{Adv}_{\mathrm{DIC}}$ corrupts only the party Rec before the protocol starts. So, the remaining steps are identical to those in the above-mentioned No Corruption Case without the message input party selection.

i. After receiving the corrupt message from Env, Sim_{DIC} prepares a situation where only Rec is corrupted adding the following policy before 1(a)iB. Sim_{DIC} makes Adv corrupt Rec.

ii. After receiving $\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}}$ from F_{DIC}, Sim_{DIC} executes the following simulation.

   A. If the message is input to corrupted party Rec, Sim_{DIC} inputs $\mathit{in}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Rec}}$ to Rec.

   B. Else the message is input to Init, and Sim_{DIC} inputs $\mathit{in}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}}$ to Init.

   C. The remaining steps are the same as those in the simulation of the No Corrupted Case.

iii. After receiving $\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}}$ in $\overline{\text{Rec}}$, Sim_{DIC} executes $\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\overline{\text{Rec}}}$.

Adv_{DIC} and Sim_{DIC} identify the direction that the message is sent, i.e., from Init to Rec or from Rec to Init. However, the simulation is perfectly executed. If the protocol executes the establish session, data sending session, and expire session, in any case, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by Sim_{DIC}. From the Task Correspondence in 7.2.1, the state correspondences in 7.1, 7.2, and 7.3 hold in this case. That is, $\mathit{trace}(\epsilon'_{\text{R}}) = \mathit{trace}(\epsilon'_{\text{I}})$ holds.

(c) **Both parties are corrupted by Adv.** This case is also analogous to case 1. This case means that Adv_{DIC} corrupts both Init and Rec before the protocol starts. So, the remaining steps are identical to those in the above-mentioned No Corruption Case without the message input party selection.

i. After receiving the corrupt message from Env, Sim_{DIC} prepares a situation in which Init and Rec are corrupted and adds the following policy before 1(a)iB. Sim_{DIC} makes Adv corrupt Init and Rec.

ii. If the data sending message is input to $\mathit{party} \in \{\mathrm{Init}, \mathrm{Rec}\}$, $\mathrm{Sim}_{\mathrm{DIC}}$ inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathit{party}}$ to *party*.

iii. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$ in $\overline{\mathit{party}}$ in his simulation, $\mathrm{Sim}_{\mathrm{DIC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathit{party}}}$.

iv. The remaining steps are the same as those in the simulation of the No Corrupted Case.

In this case, $\mathrm{Adv}_{\mathrm{DIC}}$ and $\mathrm{Sim}_{\mathrm{DIC}}$ can identify the direction that the message is sent, i.e., from Init to Rec or from Rec to Init. However, the simulation is perfectly executed. If the protocol runs the establish session, data sending session, and expire session, in any case, the simulator can simulate the real world and the movement of the Adv. That is, the simulation is perfectly executed by $\mathrm{Sim}_{\mathrm{DIC}}$. From the Task Correspondence in 7.2.1, the state correspondences in 7.1, 7.2, and 7.3 hold in this case. That is, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$ holds.

3. **Adaptive Corruption** In this case, an adversary corrupts some parties when he wants to do so at any time. We assume that the adversary corrupts the parties. However, this case is also simulated by the simulator $\mathrm{Sim}_{\mathrm{DIC}}$, so the simulation is perfectly executed. This case is separated into the following instances.

   (a) **Establish Session**

   Instance 1: Before Init and Rec are activated.
   This case is analogous to case 2 because there is no secret information in this time. The adversary can corrupt Init, Rec, or both, but the simulator can also corrupt the corresponding parties. This case is perfectly simulated by $\mathrm{Sim}_{\mathrm{DIC}}$.

   Instance 2: After Init is activated and before Rec is activated.

This case is analogous to case 2 because there is no secret information. The adversary can corrupt Init, Rec, or both, but the simulator can also corrupt the corresponding parties. This case is perfectly simulated by $\mathrm{Sim}_{\mathrm{DIC}}$.

Instance 3: After Rec is activated and before Init is activated.

This case is analogous to the case 2 because there is no secret information. The adversary can corrupt Init or Rec, or both, but the simulator can also corrupts the corresponding parties. This case is also perfectly simulated by $\mathrm{Sim}_{\mathrm{DIC}}$.

Instance 4: After Init and Rec are activated.

This case is analogous to the case 2 because there is no secret information. The adversary can corrupt Init or Rec, or both, but the simulator can also corrupt the corresponding parties. This case is perfectly simulated by $\mathrm{Sim}_{\mathrm{DIC}}$.

(b) **Data Sending Session**

Instance 1: Before or after Init or Rec is activated by receiving $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ or $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$, respectively, from the Env.

The adversary can corrupt Init, or Rec, or both, but the simulator can also corrupt the corresponding parties. This case is also perfectly simulated by $\mathrm{Sim}_{\mathrm{DIC}}$.

Env can execute only the message sending indication and the corrupt indication. So, this case represents only the case in which the adversary corrupts one party. This case is also simulated by $\mathrm{Sim}_{\mathrm{DIC}}$ because there is no secret information in this session. The task correspondence works well and there exists a simulation relation between the real world and ideal world. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$ holds.

(c) **Expire Session**

Instance 1: After Init or Rec is activated with the expire message.

Once the expire message is sent to Init or Rec by Env, This session terminates in the real world and ideal world. So the adversary can corrupt the parties. That is, this case is identical to case 2.

Simulation Policy

Sim_{DIC} executes a simulation in his simulation world as described hereafter.

(a) After receiving the "corrupt Init" message from Env,

- Sim_{DIC} corrupts $\overline{\text{Init}_{\text{DIC}}}$ and checks whether $party \in \{\text{Init}, \text{Rec}\}$ has already sent the data sending message to the other parties. If the message was already sent, Sim_{DIC} does the following. Else, Sim_{DIC} makes Adv corrupt Init.
- If $party = \text{Init}$,
  - If Sim_{DIC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\text{Init}}$ to Init in his simulation, then Sim_{DIC} simulates that Adv corrupts Init, immediately.
  - Else, Sim_{DIC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\text{Rec}}$ to Rec in his simulation, then Sim_{DIC} simulates that Adv corrupts Rec.
- Else, $party = \text{Rec}$,
  - If Sim_{DIC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\text{Init}}$ to Init in his simulation, then Sim_{DIC} simulates that Adv corrupts Rec, immediately.
  - Else, Sim_{DIC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\text{Rec}}$ to Rec in his simulation, then Sim_{DIC} simulates that Adv corrupts Init, immediately.
- If more data sending messages are input to *party* from Env after Sim_{DIC} corrupts Init, Sim_{DIC} can also simulate. If the message is input to corrupted Init, Sim_{DIC} inputs the sending message to corrupted *party* in his simulation. Else,

the message is input to Rec, and $\mathrm{Sim}_{\mathrm{DIC}}$ inputs the sending message to the non-corrupted *party* in his simulation.

- After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$ in $\overline{\mathrm{Init}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Init}}}$.

(b) After receiving the “corrupt Rec” message from Env,

- $\mathrm{Sim}_{\mathrm{DIC}}$ corrupts $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and checks whether $\mathit{party} \in \{\mathrm{Init}, \mathrm{Rec}\}$ has already sent the data sending message to the other party. If the message was already sent, $\mathrm{Sim}_{\mathrm{DIC}}$ does the following. Else, $\mathrm{Sim}_{\mathrm{DIC}}$ makes Adv corrupt Rec.
- If $\mathit{party} = \mathrm{Init}$,
  - If $\mathrm{Sim}_{\mathrm{DIC}}$ has already input message sending request $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init in his simulation, then $\mathrm{Sim}_{\mathrm{DIC}}$ simulates that Adv corrupts Rec.
  - Else, $\mathrm{Sim}_{\mathrm{DIC}}$ has already input message sending request $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec in his simulation, then $\mathrm{Sim}_{\mathrm{DIC}}$ simulates that Adv corrupts Init.
- Else, $\mathit{party} = \mathrm{Rec}$,
  - If $\mathrm{Sim}_{\mathrm{DIC}}$ has already input message sending request $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init in his simulation, then $\mathrm{Sim}_{\mathrm{DIC}}$ simulates that Adv corrupts Init, immediately.
  - Else, $\mathrm{Sim}_{\mathrm{DIC}}$ has already input message sending request $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec in his simulation, then $\mathrm{Sim}_{\mathrm{DIC}}$ simulates that Adv corrupts Rec, immediately.
- If more data sending messages are input in *party* from Env after $\mathrm{Sim}_{\mathrm{DIC}}$ corrupts Rec, $\mathrm{Sim}_{\mathrm{DIC}}$ can also simulate the real world execution. If the message is input to corrupted Init, $\mathrm{Sim}_{\mathrm{DIC}}$ inputs the sending message in the non-corrupted *party* in his simulation. Else, the message is input to Rec, and $\mathrm{Sim}_{\mathrm{DIC}}$ inputs the sending message in corrupted *party* in his simulation.

- After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\overline{\mathrm{Rec}}$, $\mathrm{Sim_{DIC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}}}$.

(c) After receiving the "corrupt Init and Rec" message from Env,

- $\mathrm{Sim_{DIC}}$ corrupts $\overline{\mathrm{Init_{DIC}}}$ and $\overline{\mathrm{Rec_{DIC}}}$ and checks whether $\mathit{party} \in \{\mathrm{Init}, \mathrm{Rec}\}$ has already sent the data sending message to the other party. If the message was already sent, $\mathrm{Sim_{DIC}}$ makes Adv corrupt Init and Rec and does the following. Else, $\mathrm{Sim_{DIC}}$ makes Adv corrupt Init and Rec.
  - If the party to which $\mathrm{Sim_{DIC}}$ has already sent a request message is equal to the party that Env sent the message, $\mathrm{Sim_{DIC}}$ inputs more data sending requests to the party.
  - Else, the input party in the simulation world is not same as the input party in the ideal world, $\mathrm{Sim_{DIC}}$ regards the input party in the simulation world as the input party which has already input a message in the ideal world. The other party in the simulation world is also regarded as the party which has not input a message yet in the ideal world.
- After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\overline{\mathit{party}}$ in his simulation, $\mathrm{Sim_{DIC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathit{party}}}$.

Whenever $\mathrm{Adv_{DIC}}$ corrupts some party, $\mathrm{Sim_{DIC}}$ corrupts the corresponding dummy party in the ideal world and forwards the obtained information to the simulated copy of $\mathrm{Adv_{DIC}}$. Conversely, $\mathrm{Sim_{DIC}}$ may obtain information from the simulated world based on the corruptions. Additionally, in this protocol the party has no secret information because $\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}$ is securely executed. In all cases, since $\mathrm{Sim_{DIC}}$ can simulate $\mathrm{Adv_{DIC}}$ using his simulated world, Env cannot distinguish the real world from the ideal world. That is, simulating party corruption is perfectly executed.

Finally, relation R is a simulation relation from the task and state correspondence with respect to the adaptive adversary. We obtain Lemma1. $\square$

Next, Theorem4 is obtained from Lemma1 immediately.

Proof. From Lemma1 and Theorem3, Theorem4 is proved. That is, the trace distribution property, $tdist(\epsilon_{\mathrm{R}}) = tdist(\epsilon_{\mathrm{I}})$ holds with respect to an adaptive adversary. As a result, the simulation is perfectly executed because $\mathrm{Sim}_{\mathrm{DIC}}$ can simulate the real world from the information message of $\mathrm{Adv}_{\mathrm{DIC}}$. The tasks of the real world perfectly correspond to the the tasks of the ideal world. That is,

$$\mathrm{Real}_{\mathrm{DIC}}||\mathrm{Env}\ \ \mathrm{Hyb.} \leq_{0}^{\mathtt{M}_{\pi_{\mathrm{DIC}}}} \mathrm{Ideal}_{\mathrm{DIC}}||\mathrm{Env}.$$

□

Functionality

(a)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Init}} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Init}_i}$
(b)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Rec}} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Rec}}$
(c)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{okcond}_{\mathrm{Adv}} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{okcond}_{\mathrm{Adv}}$
(d)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{active}$
(e)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{mes} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{mes}$
(f)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{ntask}$

Initiator

(g)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{smes} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{smes}$
(h)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{rmes} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{rmes}$
(i)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{active} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{active}$
(j)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{ntask} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{ntask}$

Receiver

(k)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.\mathit{smes} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{smes}$
(l)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.\mathit{rmes} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{rmes}$
(m)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.\mathit{active} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{active}$
(n)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.\mathit{ntask} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{ntask}$

Environment

(o)	$u.\mathrm{Env} = s.\mathrm{Env}$

Table 7.1: State Correspondence for $\mathrm{Real}_{\mathrm{DIC}}$ and $\mathrm{Ideal}_{\mathrm{DIC}}$ (Part I)

Simulator (or Adversary)

(A)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{active}$
(B)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{ntask}$
(C)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathit{smes}_X = s.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{smes}_X$
(D)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Init}_i} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Init}_i}$
(E)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Rec}} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Rec}}$
(F)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{okcond}_{\mathrm{Adv}} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{okcond}_{\mathrm{Adv}}$
(G)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{active} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{active}$
(H)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{mes} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{mes}$
(I)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{ntask} = s.\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{ntask}$
(J)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.\mathit{smes} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{smes}$
(K)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.\mathit{rmes} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{rmes}$
(L)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{active}$
(M)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Init}_{\mathrm{DIC}}.\mathit{ntask}$

Table 7.2: State Correspondence for $\mathrm{Real}_{\mathrm{DIC}}$ and $\mathrm{Ideal}_{\mathrm{DIC}}$ (Part II)

Simulator (or Adversary)

(N)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{smes} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{smes}$
(O)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{rmes} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{rmes}$
(P)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{active}$
(Q)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Rec}_{\mathrm{DIC}}.\mathit{ntask}$
(R)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{active}$
(S)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{ntask}$
(T)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{smes}_X = s.\mathrm{Adv}_{\mathrm{DIC}}.\mathit{smes}_X$

Table 7.3: State Correspondence for $\mathrm{Real}_{\mathrm{DIC}}$ and $\mathrm{Ideal}_{\mathrm{DIC}}$ (Part III)

1. **Establish Session**

(a)	$\mathrm{Init}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$ $=_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
(b)	$\mathrm{Rec}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$ $=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
(c)	$\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$

2. **Expire Session**

(a)	$\mathrm{Init}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$ $=_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{DIC}}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
(b)	$\mathrm{Rec}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$ $=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
(c)	$\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$

3. **Environment**

(a)	All tasks of environment Env in $\mathrm{Real}_{\mathrm{DIC}}$ correspond with the tasks of environment in $\mathrm{Ideal}_{\mathrm{DIC}}$.

Table 7.4: Corresponding Tasks for $\mathrm{Real}_{\mathrm{DIC}}$ and $\mathrm{Ideal}_{\mathrm{DIC}}$

Real Execution

No.	$\mathrm{Init}_{\mathrm{DIC}}$	$\mathrm{Rec}_{\mathrm{DIC}}$
1	$in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$	
2	$send(\mathtt{Send}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$	
3		
4		
5		
6		
7	$receive(\mathtt{Receive}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}}$	$receive(\mathtt{Receive}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$
8		$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Rec}}$
9		

Ideal Execution

No.	$\overline{\mathrm{Init}_{\mathrm{DIC}}}$	$\overline{\mathrm{Rec}_{\mathrm{DIC}}}$
1	$in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Init}}}$	
2	$send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$	
3		
4		
5		
6		
7		$receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
8		$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}}}$

Table 7.5: Corresponding Task Sequence of Data Sending Session for $\mathrm{Real}_{\mathrm{DIC}}$ and $\mathrm{Ideal}_{\mathrm{DIC}}$ (Part I)

Real Execution

No.	F^{I}_{2AC}	F^{R}_{2AC}	Adv_{DIC}
1			
2			
3	$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_{\mathrm{DIC}}}$	$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_{\mathrm{DIC}}}$	
4	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Adv}}$	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Adv}}$	$\mathit{receive}(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
5			$\mathit{send}(\mathtt{Response}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
6	$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{Adv}}$	$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{Adv}}$	
7	$\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Init}_{\mathrm{DIC}}}$	$\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Rec}_{\mathrm{DIC}}}$	
8			
9			
10			

Ideal Execution

No.	F_{DIC}	Sim_{DIC}
1		
2		
3	$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{X}$	
4	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$	$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
5		$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{DIC}}}$
6	$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{Adv}}$	
7	$\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\overline{X}}$	
8		

Table 7.6: Corresponding Task Sequence of Data Sending Session for $\mathrm{Real}_{\mathrm{DIC}}$ and $\mathrm{Ideal}_{\mathrm{DIC}}$ (Part II)

Real Execution

	Init$_{\text{DIC}}$								Rec$_{\text{DIC}}$							
	active		*ntask*		*smes*		*rmes*		*active*		*ntask*		*smes*		*rmes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	*m*	-	-	$\top$	-	$\perp$	DSS2	$\perp$	*m*	-	-
DSS2†	-	-	DSS2	$\perp$	-	-	-	-	-	-	DSS2	$\perp$	-	-	-	-
DSS3	$\top$	-	$\perp$	DSS4	-	$(\perp)$	$\perp$	*	$\top$	-	$\perp$	DSS4	-	$(\perp)$	$\perp$	*
DSS4	-	-	DSS4	$\perp$	-	-	-	$\perp$	-	-	DSS4	$\perp$	-	-	-	$\perp$
Initial value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	

* If *smes* = $\perp$, then *rmes* := *m* and *ntask* := DSS4.
Else *smes* := $\perp$ and *ntask* := $\perp$.

† The pre: are satisfied, then F^{I}_{2AC} and F^{R}_{2AC} are executed in random order.

Ideal Execution

	$\overline{\text{Init}_{\text{DIC}}}$								$\overline{\text{Rec}_{\text{DIC}}}$							
	active		*ntask*		*smes*		*rmes*		*active*		*ntask*		*smes*		*rmes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	*m*	-	-	$\top$	-	$\perp$	DSS2	$\perp$	*m*	-	-
DSS2	-	-	DSS2	$\perp$	-	$\perp$	-	-	-	-	DSS2	$\perp$	-	$\perp$	-	-
DSS3	$\top$	-	$\perp$	DSS4	-	-	$\perp$	*m*	$\top$	-	$\perp$	DSS4	-	-	$\perp$	*m*
DSS4	-	-	DSS4	$\perp$	-	-	-	$\perp$	-	-	DSS4	$\perp$	-	-	-	$\perp$
Initial value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	

Table 7.7: State Flow of Data Sending Session for Real$_{\text{DIC}}$ and Ideal$_{\text{DIC}}$ (Part I)

Real Execution

	F_{2AC}								Adv_{DIC}					
	active		*ntask*		*mes*		$okcond_{Adv}$		*active*		*ntask*		$smes_X$ for each X	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\bot$	DSS2	$\bot$	m	-	-	$\top$	-	$\bot$	DSS2	-	m
DSS2	-	-	DSS2	DSS3	-	-	$\bot$	-	-	-	DSS2	$\bot$	-	$\bot$
DSS3	-	-	DSS3	DSS4	-	-	-	$\top$	-	-	-	-	-	-
DSS4	-	-	DSS4	$\bot$	-	$\bot$	-	$\bot$	-	-	-	-	-	-
Initial value	$\top$		$\bot$		$\bot$		$\bot$		$\top$		$\bot$		$\bot$	
Final value	$\top$		$\bot$		$\bot$		$\bot$		$\top$		$\bot$		$\bot$	

Ideal Execution

	F_{DIC}								Sim_{DIC}					
	active		*ntask*		*mes*		$okcond_{Adv}$		*active*		*ntask*		$smes_X$ for each X	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\bot$	DSS2	$\bot$	m	-	-	$\top$	-	$\bot$	DSS2	-	m
DSS2	-	-	DSS2	DSS3	-	-	$\bot$	-	-	-	DSS2	$\bot$	-	$\bot$
DSS3	-	-	DSS3	DSS4	-	-	-	$\top$	-	-	-	-	-	-
DSS4	-	-	DSS4	$\bot$	-	$\bot$	-	$\bot$	-	-	-	-	-	-
Initial value	$\top$		$\bot$		$\bot$		$\bot$		$\top$		$\bot$		$\bot$	
Final value	$\top$		$\bot$		$\bot$		$\bot$		$\top$		$\bot$		$\bot$	

Table 7.8: State Flow of Data Sending Session for $Real_{DIC}$ and $Ideal_{DIC}$ (Part II)

Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}_{\mathrm{DIC}}$,
where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathtt{DIC}})$
$\mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathrm{Init}, \mathtt{sid}'^{\mathtt{I}}_{\mathtt{2AC}})$
$\mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathrm{Rec}, \mathtt{sid}'^{\mathtt{R}}_{\mathtt{2AC}})$

Input:	Output:
$in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$	$send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$	$send(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$receive(\mathtt{Receive}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}}$	$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Init}}$
$in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$	$send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$

State:

$smes, rmes \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$active \in \{\perp, \top\}$, initially $\perp$

Tasks:

$\{send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}, send(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}},$
$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Init}}, send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}\}$

Figure 7.10: Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}_{\mathrm{DIC}}$ (Part I)

```
Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}_{\mathrm{DIC}}$,
where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Transitions:

    Establish Session:

        ESS1. $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$
                pre: $active, ntask = \perp$
                eff: $ntask := \mathrm{ESS2}$
        ESS2. $send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
                pre: $ntask = \mathrm{ESS2}$
                eff: after all $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ finished, then $active := \top$ and
              $ntask := \perp$

    Data Sending Session:

        DSS1. $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$
                pre: $active = \top$, $smes$ and $ntask = \perp$
                eff: $smes := m$ and $ntask := \mathrm{DSS2}$
        DSS2. $send(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
                pre: $m := smes$ and $ntask = \mathrm{DSS2}$
                eff: after all $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ finished, then $ntask := \perp$
        DSS3. $receive(\mathtt{Receive}, \mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{I}}_{\mathrm{2AC}}}$
                pre: $active = \top$, $rmes$ and $ntask = \perp$
                eff: If $smes = \perp$, then $rmes := m$ and $ntask := \mathrm{DSS4}$.
                Else $smes := \perp$ and $ntask := \perp$.
        DSS4. $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Init}}$
                pre: $r := rmes$ and $ntask = \mathrm{DSS4}$
                eff: $rmes$ and $ntask := \perp$
```

Figure 7.11: Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}_{\mathrm{DIC}}$ (Part II)

```
Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}_{\mathrm{DIC}}$,
where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Transitions:

    Expire Session:
        EXS1. $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$
            pre: $active = \top$ and $ntask = \perp$
            eff: $ntask := \mathrm{EXS2}$
        EXS2. $send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
            pre: $ntask = \mathrm{EXS2}$
            eff: after all $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ finished, then $active$ and $ntask := \perp$
```

Figure 7.12: Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}_{\mathrm{DIC}}$ (Part III)

Code for Receiver of Direction-Indeterminable Channel, $\mathrm{Rec}_{\mathrm{DIC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathtt{DIC}})$
$\mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathrm{Init}, \mathtt{sid}'^{\mathtt{I}}_{\mathtt{2AC}})$
$\mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathrm{Rec}, \mathtt{sid}'^{\mathtt{R}}_{\mathtt{2AC}})$

Input:	Output:
$in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$	$send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$	$send(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$receive(\mathtt{Receive}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$	$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Rec}}$
$in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$	$send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$

State:

$smes, rmes \in \{0,1\}^* \cup \{\bot\}$, initially $\bot$ $\quad ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$
$active \in \{\bot, \top\}$, initially $\bot$

Tasks:

$\{send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}, send(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}},$
$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Rec}}, send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}\}$

Figure 7.13: Code for Receiver of Direction-Indeterminable Channel, $\mathrm{Rec}_{\mathrm{DIC}}$ (Part I)

Code for Receiver of Direction-Indeterminable Channel, $\mathrm{Rec}_{\mathrm{DIC}}$

Transitions:

```
Establish Session:
    ESS1. $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$
        pre: $active$ and $ntask = \perp$
        eff: $ntask :=$ ESS2
    ESS2. $send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
        pre: $ntask =$ ESS2
        eff: after all $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ finished, then $active := \top$ and $ntask := \perp$
Data Sending Session:
    DSS1. $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$
        pre: $active = \top$, $smes$ and $ntask = \perp$
        eff: $smes := m$ and $ntask :=$ DSS2
    DSS2. $send(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
        pre: $m := smes$ and $ntask =$ DSS2
        eff: after all $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ finished, then $ntask := \perp$
    DSS3. $receive(\mathtt{Receive}, \mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{R}}_{\mathrm{2AC}}}$
        pre: $active = \top$, $rmes$ and $ntask = \perp$
        eff: If $smes = \perp$, then $rmes := m$ and $ntask :=$ DSS4
        Else $smes := \perp$ and $ntask := \perp$
    DSS4. $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{Rec}}$
        pre: $r := rmes$ and $ntask =$ DSS4
        eff: $rmes$ and $ntask := \perp$
Expire Session:
    EXS1. $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$
        pre: $active = \top$ and $ntask = \perp$
        eff: $ntask :=$ EXS2
    EXS2. $send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
        pre: $ntask =$ EXS2
        eff: after all $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ finished, then $active$ and $ntask := \perp$
```

Figure 7.14: Code for Receiver of Direction-Indeterminable Channel, $\mathrm{Rec}_{\mathrm{DIC}}$ (Part II)

Code fot Adversary for Direction Indeterminable Channel, $\mathrm{Adv}_{\mathrm{DIC}}$, where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Signature:

$\mathtt{sid}^{\mathtt{I}}_{\mathtt{2AC}} = (\{\mathrm{Init}_1, \mathrm{Init}_2\}, \mathrm{Init}_1, \mathtt{sid}'_{\mathtt{2AC}})$
$\mathtt{sid}^{\mathtt{R}}_{\mathtt{2AC}} = (\{\mathrm{Init}_1, \mathrm{Init}_2\}, \mathrm{Init}_2, \mathtt{sid}'_{\mathtt{2AC}})$

Input:
$receive(\mathtt{SID}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$receive(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
$receive(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$

Output:
$send(\mathtt{Response}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, ok)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$

Other:
*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message and $out(*)$.

State:

$active \in \{\perp, \top\}$, initially $\perp$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$smes_X \in (\{0,1\}) \cup \{\perp\}$, initially $\perp$

Tasks:

$\{send(\mathtt{Response}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, ok)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$, other arbitrary tasks$\}$

Figure 7.15: Code fot Adversary for Direction Indeterminable Channel, $\mathrm{Adv}_{\mathrm{DIC}}$ (Part I)

Code fot Adversary for Direction Indeterminable Channel, $\mathrm{Adv}_{\mathrm{DIC}}$,
where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$

Transitions:

```
Establish Session:
    ESS1. $receive(\mathtt{SID}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
        pre: $active = \bot$
        eff:$active := \top$
Data Sending Session:
    DSS1. $receive(\mathtt{Send}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
        pre: $active = \top$ and $ntask = \bot$
        eff: $smes_X := m$ and $ntask :=$ DSS2
    DSS2. $send(\mathtt{Response}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}}, ok)_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
        pre: $ntask =$ DSS2
        eff: $smes_X, ntask := \bot$
Expire Session:
    EXS1. $receive(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{\mathrm{2AC}}}$
        pre: $active = \top$
        eff: $active := \bot$
Other tasks:
    This adversary makes other arbitary tasks.
```

Figure 7.16: Code fot Adversary for Direction Indeterminable Channel, $\mathrm{Adv}_{\mathrm{DIC}}$ (Part II)

Code for ideal Initiator of Direction-Indeterminable Channel, $\overline{\text{Init}}_{\text{DIC}}$

Signature:

$\texttt{sid}_{\texttt{DIC}} = (\{\overline{\text{Init}}, \overline{\text{Rec}}\}, \texttt{sid}'_{\texttt{DIC}})$

Input:	Output:
$\mathit{in}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\overline{\text{Init}}}$	$\mathit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
$\mathit{in}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\overline{\text{Init}}}$	$\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}}$
$\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, \mathit{mes})_{\text{F}_{\text{DIC}}}$	$\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, \mathit{mes})_{\overline{\text{Init}}}$
$\mathit{in}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\overline{\text{Init}}}$	$\mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$

State:

$\mathit{smes}, \mathit{rmes} \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$ $\quad$ $\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$

$\mathit{active} \in \{\perp, \top\}$, initially $\perp$

Tasks:

$\{\mathit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}, \mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}},$
$\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, \mathit{mes})_{\overline{\text{Init}}}, \mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}\}$

Figure 7.17: Code for Initiator of Direction-Indeterminable Channel, $\overline{\text{Init}}_{\text{DIC}}$ (Part I)

```
Code for ideal Initiator of Direction-Indeterminable Channel, $\overline{\text{Init}}_{\text{DIC}}$

Transitions:

    Establish Session:

        ESS1. $in(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\overline{\text{Init}}}$
            pre: $active, ntask = \bot$
            eff: $ntask := \text{ESS2}$
        ESS2. $send(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
            pre: $ntask = \text{ESS2}$
            eff: $active := \top$ and $ntask := \bot$

    Data Sending Session:

        DSS1. $in(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\overline{\text{Init}}}$
            pre: $active = \top$, $smes$ and $ntask = \bot$
            eff: $smes := m$ and $ntask := \text{DSS2}$
        DSS2. $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}}$
            pre: $m := smes$ and $ntask = \text{DSS2}$
            eff: $smes$ and $ntask := \bot$
        DSS3. $receive(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}}$
            pre: $active = \top$, $rmes$ and $ntask = \bot$
            eff: $rmes := m$ and $ntask := \text{DSS4}$
        DSS4. $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\overline{\text{Init}}}$
            pre: $m := rmes$ and $ntask = \text{DSS4}$
            eff: $rmes$ and $ntask := \bot$

    Expire Session:

        EXS1. $in(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\overline{\text{Init}}}$
            pre: $active = \top$ and $ntask = \bot$
            eff:$ntask := \text{EXS2}$
        EXS2. $send(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
            pre: $ntask = \text{EXS2}$
            eff: $active$ and $ntask := \bot$
```

Figure 7.18: Code for Initiator of Direction-Indeterminable Channel, $\overline{\text{Init}}_{\text{DIC}}$ (Part II)

Code for ideal Receiver of Direction-Indeterminable Channel, $\overline{\mathrm{Rec}}_{\mathrm{DIC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\overline{\mathrm{Init}}, \overline{\mathrm{Rec}}\}, \mathtt{sid}'_{\mathtt{DIC}})$

Input:	Output:
$\mathit{in}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$	$\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}}}$	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\mathrm{F}_{\mathrm{DIC}}}$	$\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\overline{\mathrm{Rec}}}$
$\mathit{in}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$	$\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

State:

$\mathit{smes}, \mathit{rmes} \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$ $\quad$ $\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$

$\mathit{active} \in \{\perp, \top\}$, initially $\perp$

Tasks:

$\{\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}, \mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}},$
$\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\overline{\mathrm{Rec}}}, \mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}\}$

Figure 7.19: Code for ideal Receiver of Direction-Indeterminable Channel, $\overline{\mathrm{Rec}}_{\mathrm{DIC}}$ (Part I)

Code for ideal Receiver of Direction-Indeterminable Channel, $\overline{\mathrm{Rec}}_{\mathrm{DIC}}$

Transitions:

Establish Session:

```
ESS1. $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$
        pre: $active$ and $ntask = \perp$
        eff: $ntask :=$ ESS2
ESS2. $send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $ntask =$ ESS2
        eff: $active := \top$ and $ntask := \perp$
```

Data Sending Session:

```
DSS1. $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}}}$
        pre:$active = \top$, $smes$ and $ntask = \perp$
        eff: $smes := m$ and $ntask :=$ DSS2
DSS2. $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $m := smes$ and $ntask =$ DSS2
        eff: $smes$ and $ntask := \perp$
DSS3. $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $rmes = \perp$ and $ntask = \perp$
        eff: $rmes := m$ and $ntask :=$ DSS4
DSS4. $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}}}$
        pre: $m := rmes$ and $ntask =$ DSS4
        eff: $rmes$ and $ntask := \perp$
```

Expire Session:

```
EXS1. $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$
        pre: $active = \top$, $smes, rmes$ and $ntask = \perp$
        eff:$ntask :=$ EXS2
EXS2. $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $ntask =$ EXS2
        eff: $active$ and $ntask := \perp$
```

Figure 7.20: Code for ideal Receiver of Direction-Indeterminable Channel, $\overline{\mathrm{Rec}}_{\mathrm{DIC}}$ (Part II)

Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\overline{\mathrm{Init}}, \overline{\mathrm{Rec}}\}, \mathtt{sid}'_{\mathtt{DIC}})$

Input:
$\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F_{DIC}}}$
$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$
$\mathit{receive}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F_{DIC}}}$

Output:
$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F_{DIC}}}$

Other:
*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message and $\mathit{out}(*)$.

State:

$\mathit{active} \in \{\perp, \top\}$, initially $\perp$ $\qquad$ $\mathit{smes} \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$
$\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$

Other arbitrary variables; call ”new” variables.

Tasks:

$\{\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F_{DIC}}}\}$

Figure 7.21: Code fot Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$ (PartI)

Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$

Transitions:

Establish Session:

```
ESS1. $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $active$ and $ntask = \bot$
        eff: $active := \top$
```

Data Sending Session:

```
DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $active = \top$ and $ntask = \bot$
        eff: $smes := m$ and $ntask :=$ DSS2
DSS2. $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $ntask =$ DSS2
        eff: $smes_X, ntask := \bot$
```

Expire Session:

```
EXS1. $receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $active = \top$
        eff: $active := \bot$
```

Other tasks:

This simulator makes arbitrary tasks to simulate the real world protocol system $\mathrm{Real}_{\mathrm{DIC}}$. The tasks mey be run with the information obtained from the simulator. Additionaly, this simulator can output the message from the adversary of the simiulating world to the environment.

Figure 7.22: Code fot Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$ (PartII)

7.2.2 Reduction of 2AC to DIC

Let $\mathrm{Init}_{\mathrm{2AC_i}}$ ($i \in \{1,2\}$) and $\mathrm{Rec}_{\mathrm{2AC}}$ be the initiator code and receiver code for a real system, see Fig.7.23 and Fig.7.24, and Fig.7.25 and Fig.7.26, respectively. Let $\overline{\mathrm{Init}_{\mathrm{2AC_i}}}$ and $\overline{\mathrm{Rec}_{\mathrm{2AC}}}$ be the initiator code and receiver code for an ideal system, see Fig.7.29 and Fig.7.30, and Fig.7.31 and Fig.7.32, respectively. Finally, let $\mathrm{Adv}_{\mathrm{2AC}}$ and $\mathrm{Sim}_{\mathrm{2AC}}$ be the adversary code and the simulator code in Fig.7.27 and Fig.7.28, and Fig.7.33 and Fig.7.34, respectively. Let $\mathrm{Real}_{\mathrm{2AC}}$ and $\mathrm{Ideal}_{\mathrm{2AC}}$ be a 2AC protocol system and a 2AC functionality system, respectively, defined as described hereafter.

$$\mathrm{Real}_{\mathrm{2AC}} := \mathrm{Init}_{\mathrm{2AC_i}} || \mathrm{Rec}_{\mathrm{2AC}} || \mathrm{Adv}_{\mathrm{2AC}} || \mathrm{F}_{\mathrm{DIC}},$$
$$\mathrm{Ideal}_{\mathrm{2AC}} := \overline{\mathrm{Init}_{\mathrm{2AC_i}}} || \overline{\mathrm{Rec}_{\mathrm{2AC}}} || \mathrm{Sim}_{\mathrm{2AC}} || \mathrm{F}_{\mathrm{2AC}}.$$

Tasks $\overline{\mathrm{Init}_{\mathrm{2AC_i}}}$ and $\overline{\mathrm{Rec}_{\mathrm{2AC}}}$ relay the input messages from the environment to the ideal functionality task and relay the messages received from the ideal functionality task to the environment as interface parties in the ideal system.

Master Schedule

Let n be the number of parties. Let $M_{psync}(t_1^*, \cdots, t_n^*)$ be master schedules where t_i^* is a task in party P_i.

Definition 25. [$M_{psync}(t_1^*, \cdots, t_n^*)$] *Let t_i^* be a task in party P_i. Let $ptask(t_i^*)$ be the task just before t_i^* in local schedule ρ_i. For example, let $\rho_i = t_{i1}, t_{i2}, t_{i3}$ for party P_i. Then $ptask(t_{i3})$ is task t_{i2}.*

1. *Alignment property: After master schedule M activates $ptask(t_i^*)$, M does not activate P_i until all of $ptask(t_1^*), \cdots, ptask(t_n^*)$ are scheduled. This situation indicates that M satisfies the alignment property for specified tasks $t_1^*, \ldots, t_n^*$.*

2. *Random execution property: The master schedule, M, globally executes specified tasks, $t_1^*, \ldots, t_n^*$ in a random order. Note that the other tasks are not scheduled until all of the specified tasks, $t_1^*, \ldots, t_n^*$, are executed.*

$M_{psync}(t_1^*, \cdots, t_n^*)$ is defined to be a master schedule such that a master schedule M satisfies the two above-mentioned properties for specified tasks $t_1^*, \ldots, t_n^*$. Let $M_{\pi_{2\mathrm{AC}}}$ be $M_{psync}(\mathrm{Init}^1_{2\mathrm{AC}}.\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, s)_{\mathrm{Rec}_{2\mathrm{AC}}},$ $\mathrm{Init}^2_{2\mathrm{AC}}.\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, s)_{\mathrm{Rec}_{2\mathrm{AC}}})$ for $\pi_{2\mathrm{AC}}$.

Theorem 5. *2AC protocol system* $\mathrm{Real}_{2\mathrm{AC}}$ *perfectly hybrid-implements 2AC functionality system* $\mathrm{Ideal}_{2\mathrm{AC}}$ *with respect to an adaptive adversary under* $M_{\pi_{2\mathrm{AC}}}$ *(2AC is reducible to DIC with respect to an adaptive adversary under* $M_{\pi_{2\mathrm{AC}}}$*).*

Let ϵ_{R} and ϵ_{I} be discrete probability measures on finite executions of $\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}$ and $\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}$, respectively. We prove Theorem 5 by showing that ϵ_{R} and ϵ_{I} satisfy the trace distribution property, $\mathit{tdist}(\epsilon_{\mathrm{R}}) = \mathit{tdist}(\epsilon_{\mathrm{I}})$. Here, we define correspondence R between the states in $\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}$ and the states in $\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}$. We say $(\epsilon_{\mathrm{R}}, \epsilon_{\mathrm{I}}) \in R$ if and only if for every $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$, all of the state correspondences in Tables 7.9, 7.10 and 7.11 hold. We then prove R is a simulation relation in Lemma 2.

Lemma 2. *Relation* R *defined above is a simulation relation from* $\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}$ *to* $\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}$ *under master schedule* $M_{\pi_{2\mathrm{AC}}}$.

Proof. We prove that R is a simulation relation from $\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}$ to $\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}$ using mapping corrtasks $R^*_{\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}} \times R_{\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}} \rightarrow R^*_{\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}}$.

The task sequence of system $\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}$ perfectly corresponds to that of system $\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}$ under schedule $M_{\pi_{2\mathrm{AC}}}$. Formally, to prove that R is a simulation relation from $\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}$ to $\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}$, we show that R satisfies the start condition and step condition.

- **Start condition**
  It is true that the start states of s and u in $\mathrm{Real}_{2\mathrm{AC}}||\mathrm{Env}$ and $\mathrm{Ideal}_{2\mathrm{AC}}||\mathrm{Env}$, respectively, are on the Dirac measures. That is, the start states of s and u satisfy relation R because the start states of s and u are all $\perp$ for each task on master schedule $M_{\pi_{\mathrm{DIC}}}$. Therefore, the trace distribution property holds.

- **Step condition**
  Let $\epsilon'_{\mathrm{R}} = apply(\epsilon_{\mathrm{R}}, T)$ and $\epsilon'_{\mathrm{I}} = apply(\epsilon_{\mathrm{I}}, corrtasks(\rho, T))$. If $(\epsilon_{\mathrm{R}}, \epsilon_{\mathrm{I}}) \in R$, $\rho \in R^*_{\mathrm{Real_{2AC}}||\mathrm{Env}}$, ϵ_{R} is consistent with ρ, then ϵ_{I} is consistent with $full(corrtasks)(\rho)$, and $T \in \mathrm{Real_{2AC}}||\mathrm{Env}$. Then there exist the following.

  - Probability measure p on countable index set I,
  - Probability measures $\epsilon'_{\mathrm{R},j}$, $j \in I$, on finite executions of $\mathrm{Real_{2AC}}||\mathrm{Env}$, and
  - Probability measures $\epsilon'_{\mathrm{I},j}$, $j \in I$, on finite executions of $\mathrm{Ideal_{2AC}}||\mathrm{Env}$,

  such that:

  - For each $j \in I$, $\epsilon'_{\mathrm{R},j} \; R \; \epsilon'_{\mathrm{I},j}$,
  - $\Sigma_{j \in I} p(j)(\epsilon'_{\mathrm{R},j}) = apply(\epsilon_{\mathrm{R}}, T)$, and
  - $\Sigma_{j \in I} p(j)(\epsilon'_{\mathrm{I},j}) = apply(\epsilon_{\mathrm{I}}, corrtask(\rho, T))$.

Task Correspondence

For any $(\rho, T) \in (R^*_{\mathrm{Real_{2AC}}||\mathrm{Env}} \times R_{\mathrm{Real_{2AC}}||\mathrm{Env}})$, the following task correspondence, which is also summarized in Table 7.12, holds.

1. **Establish Session**

   (a) $\mathrm{Init_{2AC_i}}.send(\mathtt{Establish_{DIC}}, \mathtt{sid_{DIC}})_{\mathrm{F^{(1,2)}_{DIC}}}$
   $=_{\mathrm{corr.}} \overline{\mathrm{Init_{2AC_i}}}.send(\mathtt{Establish_{2AC}}, \mathtt{sid_{2AC}})_{\mathrm{F_{2AC}}}$ for each $i \in \{1, 2\}$

   Let $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ be $send(\mathtt{Establish_{DIC}}, \mathtt{sid_{DIC}})_{\mathrm{F^{(1,2)}_{DIC}}}$ and $send(\mathtt{Establish_{2AC}}, \mathtt{sid_{2AC}})_{\mathrm{F_{2AC}}}$, respectively. Here, we must consider the cases of $\mathrm{Init^1_{2AC}}$ and $\mathrm{Init^2_{2AC}}$, but these cases follow the same discussion. (Note that we describe that $\mathrm{Init^1_{2AC}}$ means $\mathrm{Init_{2AC_1}}$ and that $\mathrm{Init^2_{2AC}}$ means $\mathrm{Init_{2AC_2}}$ for the sake of clarity).

So, we consider the case of $\mathrm{Init}^1_{2\mathrm{AC}}$. We assume that for each state in $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. The preconditions of $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are $ntask = \mathrm{ESS2}$ from each code. $\mathtt{T}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is enabled (or disabled) in s (resp., u) if and only if $s.\overline{\mathrm{Init}^1_{2\mathrm{AC}}}.ntask = \mathrm{ESS2}$ (resp. $u.\overline{\mathrm{Init}^1_{2\mathrm{AC}}}.ntask = \mathrm{ESS2}$). From ($i$) in Table 7.9, $u.\mathrm{Init}^1_{2\mathrm{AC}}.ntask$ and $s.\mathrm{Init}^1_{2\mathrm{AC}}.ntask$ imply that $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

i. Disable Case:
   Let I and p be the set that has a single element and Dirac measure on I, respectively. Let $\epsilon'_{\mathrm{R},1} = \epsilon'_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon'_{\mathrm{I}}$. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, we obtain $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ from relation $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. The trace distribution equivalence property, $tdist(\epsilon'_{\mathrm{R}}) = tdist(\epsilon'_{\mathrm{I}})$, also holds since $tdist(\epsilon_{\mathrm{R}}) = tdist(\epsilon_{\mathrm{I}})$ under $M_{\pi_{2\mathrm{AC}}}$.

ii. Enable Case:
   Let q denote the state of precondition : $ntask = \mathrm{ESS2}$. Let $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ be the actions enabled in q in each world. We show that each of $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ is a unique action that is enabled in q. From the definition of $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$, the precondition is only $ntask = \mathrm{ESS2}$. Then, there are two unique effects that update the *active* and *ntask* to be $\top$ and $\perp$, respectively. From the precondition and the effect of $\mathtt{T}_{\mathtt{REAL}}$, and the state equivalence of (`h`) and (`i`), we obtain that $\mathtt{T}_{\mathtt{REAL}}$ (and $\mathtt{T}_{\mathtt{IDEAL}}$) is also a unique action that is enabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.
   Let I and p be the set that has a single element and the Dirac measure on I, respectively. Let $\epsilon'_{\mathrm{R},1} = \epsilon'_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon'_{\mathrm{I}}$. Here, we establish the property of R for ϵ'_{R} and ϵ'_{I} to show that $(\epsilon'_{\mathrm{R}}, \epsilon'_{\mathrm{I}}) \in R$. Then we show trace distribution equivalence for ϵ'_{R} and ϵ'_{I}. To establish this property, we consider any state $s' \in \mathrm{supp.lst}(\epsilon'_{\mathrm{R}})$ and $u' \in \mathrm{supp.lst}(\epsilon'_{\mathrm{I}})$. Let s be any

state in supp.lst(ϵ_R) such that $s' \in \text{supp}(\mu_s)$, where $(s, \zeta, \mu_s) \in \text{Real}_{2AC}||\text{Env}$. Let u be any state in supp.lst(ϵ_I) such that $u' \in \text{supp}(\mu_u)$ where $(u, corrtask(\rho, \zeta), \mu_u) \in \text{Ideal}_{2AC}||\text{Env}$. It is true that $\mathtt{T_{REAL}}$ updates $\text{Init}^1_{2AC}.active$ to $\top$ and $\text{Init}^1_{2AC}.ntask$ to $\bot$ from the definition of the effect of $\mathtt{T_{REAL}}$. Similarly, $\mathtt{T_{IDEAL}}$ updates $\overline{\text{Init}}^1_{2AC}.active$ to $\top$ and $\overline{\text{Init}}^1_{2AC}.ntask$ to $\bot$ from the definition of the effect of $\mathtt{T_{IDEAL}}$. From the state equivalences of (h) and (i) in Table 7.9, we have $u.\overline{\text{Init}}^1_{2AC}.active = s.\text{Init}^1_{2AC}.active$ and $u.\overline{\text{Init}}^1_{2AC}.ntask = s.\text{Init}^1_{2AC}.ntask$. We obtain that $u'.\overline{\text{Init}}^1_{2AC}.active = s'.\text{Init}^1_{2AC}.active$ and $u'.\overline{\text{Init}}^1_{2AC}.ntask = s'.\text{Init}^1_{2AC}.ntask$. By the definition of Init^1_{2AC} and $\overline{\text{Init}}^1_{2AC}$, $\mathtt{T_{REAL}}$ (resp., $\mathtt{T_{IDEAL}}$) is a unique action that updates the state of $active$ of Real_{2AC} (resp., Ideal_{2AC}). Therefore, we obtain trace distribution property $trace(\epsilon'_R) = trace(\epsilon'_I)$.

The case of Init^2_{2AC} (and $\overline{\text{Init}}^2_{2AC}$) follows the same discussion as that above.

(b) $\text{F}_{\text{DIC}}.send(\mathtt{SID}, \mathtt{sid_{DIC}})_{\text{Adv}} =_{\text{corr.}} \text{F}_{2AC}.send(\mathtt{SID}, \mathtt{sid_{2AC}})_{\text{Adv}}$

The precondition and effect of these tasks are identical to each other. The preconditions for the task on the left side of the equation is $active = \top$ and $ntask = \text{ESS2}$. This is equivalent to those for the task on the right. The effect of task on the left is $ntask := \bot$. This effect is also the same as that for the task on the right. Let $\mathtt{T_{REAL}}$ be $\text{F}_{\text{DIC}}.send(\mathtt{SID}, \mathtt{sid_{DIC}})_{\text{Adv}}$. Let $\mathtt{T_{IDEAL}}$ be $\text{F}_{2AC}.send(\mathtt{SID}, \mathtt{sid_{2AC}})_{\text{Adv}}$. We show that $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ are uniformly enabled or disabled in $\text{supp.lst}(\epsilon_R) \cup \text{supp.lst}(\epsilon_I)$. We assume that for each state in $s \in \text{supp.lst}(\epsilon_R)$ and $u \in \text{supp.lst}(\epsilon_I)$ are fixed. Then, $\mathtt{T_{REAL}}$ is enabled (or disabled) in s if and only if $s.\mathtt{T_{REAL}}.active = \top$ and $s.\mathtt{T_{REAL}}.ntask = \text{ESS2}$. The preconditions of $\mathtt{T_{IDEAL}}$, (d) and (f) in Table 7.9, imply that $\mathtt{T_{IDEAL}}$ is uniformly enabled or disabled. The rest of this proof is similar to the task of $send(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$ of the initiator.

More specifically, the pre:(d) and (f) and eff:(f) of the real task are the same as those for the ideal task. So, these tasks correspond.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondences of pre: (d) and (f). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definitions, the state correspondences of pre: (d) and (f), and state correspondence of eff: (f), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

2. **Data Sending Session**

Here, we consider the case that Env sends the data sending message in $\mathrm{Init}^1_{2\mathrm{AC}}$. The case that Env sends the data sending message in $\mathrm{Init}^2_{2\mathrm{AC}}$ is analogous to the case of $\mathrm{Init}^1_{2\mathrm{AC}}$. The task sequence in each world is shown in Table 7.13. The task sequence for Real Execution are corresponds to that for Ideal Execution.

The flow of the states for each task is shown in Tables 7.14, 7.15, and 7.16 in each world. From the initial values and final values in Tables 7.14, 7.15 and 7.16, we obtain the results of state equivalence in 7.9. That is, if the state equivalence in 7.9 holds before $\mathrm{Real}_{2\mathrm{AC}}$ is enabled (or disabled), the state equivalence in 7.9 also holds after $\mathrm{Real}_{2\mathrm{AC}}$ is enabled/disabled. The *dummy* state does not use in the ideal world's tasks, but $\mathrm{Sim}'_{\mathrm{DIC}}$ can simulate the real world in his simulation world.

The *dummy* state is not used in the tasks in the ideal world, but $\mathrm{Sim}'_{\mathrm{DIC}}$ can simulate the real world in his simulation world. So, the trace distribution property holds.

(a) Disable Case: This is a trivial case because all the states of the parties are $\perp$. The states do not change before or after the protocol starts in each world. That is, Env inputs no message to the parties. Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from the task definition and the state correspondence. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(b) Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definition and according to the states in Tables 7.14, 7.15, and 7.16, it is clear that the initial state is the same as the final state for each task of each world. In addition, the states of the real task are the same as the states of the ideal world after the data sending session is executed. That is, (a) ~ (l) in Table 7.9 hold. Therefore, $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

3. **Expire Session**

(a) $\mathrm{Init}_{\mathrm{2AC_i}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{2AC_i}}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F_{2AC}}}$
The states of precondition and effect for $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$ are the same as those for $send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F_{2AC}}}$ where pre:*ntask* = EXS2. That is, if (i) in Table 7.9 holds, then these tasks are enabled (or disabled) in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. More specifically, the pre:(i) and

eff:(h) and (i) of the real task are the same as those for the ideal task. So, these tasks correspond.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (i). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondences of pre: (i) and (l), and state correspondences of eff: (h) and (i), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(b) $\mathrm{F}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$
$=_{\mathrm{corr.}} \mathrm{F}_{\mathrm{2AC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}$
The precondition and effect for the real task are the same as those for the ideal task. The precondition is only $ntask = \mathrm{EXS2}$ and the effects are $active := \perp$ and $estcond_X := \perp$ for all X and $ntask := \perp$. From (f) in Table 7.9 these tasks are enabled (or disabled) in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. More specifically, the pre:(f) and eff:(a), (b), and (f) for the real task are the same as those for the ideal task. So, these tasks correspond.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (f). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondence of pre: (f), and state correspondences of eff: (a), (b) and (f), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Environment Env

From the task definitions and state correspondence (o) in Table7.9, the provability measures for both tasks are uniformly enabled or disabled in every state in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}).

Claim 1 The state of Env remains static in all states in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}). Let q_e denote this state of Env. This follows from state correspondence o.

Claim 2 If T is a task of Env, then T is either enabled or disabled in every state in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}) (simultaneously). Furthermore, if T is enabled in all states in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}), then:

1. There exists unique action $a \in$ T that is enabled in every state in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}).
2. There exists a unique transition of Env from q_e with action a. Let $tr_e = (q_e, a, \mu_e)$ be this transition.

By considering Claim 7.2.2, task T of Env is uniformly enabled or disabled in every state in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}). If T is disabled, let $I = 1$, we obtain $\epsilon'_{\mathrm{R},1} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon_{\mathrm{I}}$, and the result is $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ since we have $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. If T is enabled in every state in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}), Claim 7.2.2 implies that there exists unique action a in every state in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}) and transition tr_e of Env from q_e enabled with action a where $tr_e = (q_e, a, \mu_e)$.

Non Corrupted Case:

1. a is an input action of Init_i. We assume that a is an input action such as $in(\mathtt{Establish_{2AC}}, \mathtt{sid_{2AC}})_{\mathrm{Init}_i}$ $in(\mathtt{Send}, \mathtt{sid_{2AC}}, m)_{\mathrm{Init}_i}$ and $in(\mathtt{Expire_{2AC}}, \mathtt{sid_{2AC}})_{\mathrm{Init}_i}$.

   Let s be any state such that $s' \in \mathrm{supp}(\mu_\mathrm{s})$ where $(s, a, \mu_s) \in D_{\mathrm{Real_{2AC}}\|\mathrm{Env}}$. Let u be any state such that $u' \in \mathrm{supp}(\mu_\mathrm{u})$ where $(u, a, \mu_u) \in D_{\mathrm{Ideal_{2AC}}\|\mathrm{Env}}$. For each a, we check that the state correspondences for s' and u' holds if that for s and u holds. If each a is input from Env, then the precondition and effect for the real task are exactly the same as those for the ideal task. For example, if the input message is $in(\mathtt{Establish_{2AC}}, \mathtt{sid_{2AC}})_{\mathrm{Init}_i}$, then the precondition is $active, ntask = \perp$ and the effect is $ntask :=$ ESS2. The states for the real task correspond to those for the ideal task. That is, the state correspondences of (m), (h), and (i) for s' and u' hold, if the state correspondences for s and u hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$. This result also works well in the case of $in(\mathtt{Send}, \mathtt{sid_{2AC}}, m)_{\mathrm{Init}_i}$ and $in(\mathtt{Expire_{2AC}}, \mathtt{sid_{2AC}})_{\mathrm{Init}_i}$ for each $i \in \{1, 2\}$.

2. a is an input / output action of Rec. We assume that a is an input action such as $in(\mathtt{Establish_{2AC}}, \mathtt{sid_{2AC}})_{\mathrm{Rec}}$ and $out(\mathtt{Receive}, \mathtt{sid_{2AC}}, m)_{\mathrm{Rec}}$. This is analogous to case 1.

3. a is an input action of Adv. This means that $a = input(g)_{\mathrm{Adv}}$ for some fixed g. For example, g is a corrupt message for some $party \in \{\mathrm{Init}_i, \mathrm{Rec}\}$. From the fact that the state correspondences for $(A) \sim (R)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

4. a is an internal or an output action of Env. Task a in the real world is identical to that in the ideal world. From the fact that the state correspondence to (m) for s and u holds, we obtain that the state correspondence of (m) for s' and u' holds. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R},j}) = trace(\epsilon'_{\mathrm{I},j})$.

Corrupted Case:

1. a is an input action of Adv and $party \in \{\mathrm{Init}_i, \mathrm{Rec}\}$ for each $i \in \{1,2\}$ Here, the party is included in the case of $\mathrm{Init}_1 \wedge \mathrm{Rec}$, $\mathrm{Init}_2 \wedge \mathrm{Rec}$ and $\mathrm{Init}_1 \wedge \mathrm{Init}_2$. Let q_{Adv} be the state of Adv or Sim, which is the same for all $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. Let $tr_{\mathrm{Adv}} = (q_{\mathrm{Adv}}, a, \mu_{\mathrm{Adv}})$ be a transition of Adv with action a from q_{Adv}. From Claim 7.2.2, tr_{Adv} is a unique transition. Here, we suppose that $\mathrm{supp}((\mu_{\mathrm{e}} \times \mu_{\mathrm{Adv}}))$ is the pair set $\{(q_{1,j}, q_{2,j}) : j \in I\}$ where I is a countable set. Let p be the probability measures such that for each j, $p(j) = (\mu_e \times \mu_{\mathrm{Adv}})(q_{1,j}, q_{2,j})$. For each j, let $\epsilon'_{\mathrm{R},j}$ be $\epsilon'_{1,j}(\alpha) = \epsilon_1(\alpha')$ where $\alpha \in \mathrm{supp}(\epsilon'_1)$ such that $\mathtt{lst}(\alpha).\mathrm{Env} = q_{1,j}$ and $\mathtt{lst}(\alpha).\mathrm{Adv} = q_{2,j}$. The $\epsilon'_{2,j}$ is analogously constructed from ϵ'_2.

   The rest of this proof is the same as that for case 1 by considering the state correspondence in each case of $party \in \{\mathrm{Init}_i, \mathrm{Rec}, \mathrm{Init}_i \wedge \mathrm{Rec}\}$. Finally, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R},j}) = trace(\epsilon'_{\mathrm{I},j})$.

Adversary Adv

From the task definitions and the state correspondences for $(A) \sim (R)$ in Table 7.10, the provability measures for both tasks are uniformly enabled or disabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

> **Claim 3** The state of Adv or Sim is the same in all states in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. Let q_{Adv} denote this state of Adv and Sim. This follows from state correspondence of Sim.
>
> **Claim 4** If T is a task of Adv, then T is either enabled or disabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. Furthermore, if T is enabled in all states in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$, then:
>
> 1. There is unique action $a \in \mathrm{T}$ that is enabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

2. There is a unique transition of Adv from q_{Adv} with action a and let $tr_{\text{Adv}} = (q_{\text{Adv}}, a, \mu_{\text{Adv}})$ be this transition.

By considering Claim 7.2.2, task T of Adv is uniformly enabled or disabled in every state in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}). If T is disabled, let $I = 1$, we obtain $\epsilon'_{\text{R},1} = \epsilon_{\text{R}}$ and $\epsilon'_{\text{I},1} = \epsilon_{\text{I}}$, and the result is $\epsilon'_{\text{R},1} R \epsilon'_{\text{I},1}$ since we have $\epsilon_{\text{R}} R \epsilon_{\text{I}}$. If T is enabled, T is enabled in every state in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}). Claim 7.2.2 implies that there is unique action a in every state in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}) and transition tr of Adv from q_e enabled with action a where $tr_{\text{Adv}} = (q_{\text{Adv}}, a, \mu_{\text{Adv}})$. The following cases of “Non Corrupted Case” and “Corrupted Case” can be considered.

Non Corrupted Case:

1. a is an input action of Env. From the fact that the state correspondences $(A) \sim (R)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\text{R}}) = trace(\epsilon'_{\text{I}})$.

2. a is an input or output action of functionality task. This case concerns messages $receive(\texttt{SID}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$, $receive(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}}$, $receive(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$, and $send(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{F}_{\text{DIC}}}$. The rest of this proof is analogous to case 1. From the fact that the state correspondences $(A) \sim (R)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property: $trace(\epsilon'_{\text{R}}) = trace(\epsilon'_{\text{I}})$.

3. a is either an output action of Adv that is not an input action of $\text{Env}, \text{Init}_i$, Rec or functionality task, or is an internal action of Adv. This case concerns “new” tasks. The rest of this proof is analogous to case 1. From the fact that the state correspondences $(A) \sim (R)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\text{R}}) = trace(\epsilon'_{\text{I}})$.

4. a is an output action of $out(*)_{adv}$. This case is also works well although this action may effect Env. However, the transition of Env $tr_e = (q_e, a, \mu_e)$ is unique from Claim 7.2.2. Claim 7.2.2 also says that the state of Env remains static in all states in supp.lst(ϵ_{R})$\cup$supp.lst(ϵ_{I}). This follows from state correspondence o. Similarly, from the definition and some claims, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

a is an output action of Init_i. This case concerns message $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\mathrm{Rec}}$ for each i. The rest of this proof is analogous to case 1. From the fact that the state correspondences for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

a is an input action of Rec. This case concerns message $receive(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{Init}_i}$. The rest of this proof is analogous to case 1. From the fact that the state correspondences for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Corrupted Case:

This is the case in which the static and adaptive adversary, Adv, corrupts $party \in \{\mathrm{Init}_i, \mathrm{Rec}\}$.

1. a is an input/output action $in(*)_{party}$, $out(*)_{party}$ of corrupted party, $party \in \{\mathrm{Init}_i, \mathrm{Rec}\}$. This case also works well based on Claim 7.2.2 and the state correspondences in Table 7.9 $\sim$ 7.11.

Perfect Simulation

Another task of $\mathrm{Sim}_{\mathrm{2AC}}$ is the $simulation(*)$ task. By using $simulation(*)$ effectively, the simulation of $\mathrm{Sim}_{\mathrm{2AC}}$ is perfectly executed for the establish session, data sending session and expire session with respect to no corruption, static corruption, and adaptive corruption by an adversary.

1. **No Corruption**

(a) **Establish Session** First, in the establish session, Env sends $in(\mathtt{Establish}_{\mathtt{2AC}},\mathtt{sid}_{\mathtt{2AC}})_{\overline{\mathrm{Init}_i}}$ and $in(\mathtt{Establish}_{\mathtt{2AC}},\mathtt{sid}_{\mathtt{2AC}})_{\overline{\mathrm{Rec}}})$ to initiator $\overline{\mathrm{Init}_{\mathrm{2AC_i}}}$ and receiver $\overline{\mathrm{Rec}_{\mathrm{2AC}}}$, respectively. $\overline{\mathrm{Init}^1_{\mathrm{2AC}}}$ and $\overline{\mathrm{Init}^2_{\mathrm{2AC}}}$ send $send(\mathtt{Establish}_{\mathtt{2AC}},\mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F_{2AC}}}$ to $\mathrm{F_{2AC}}$. $\mathrm{F_{2AC}}$ sends $send(\mathtt{SID},\mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}$ to $\mathrm{Sim_{2AC}}$. After $\mathrm{Sim_{2AC}}$ receives the message, $\mathrm{Sim_{2AC}}$ generates parties $\mathrm{Init}^1_{\mathrm{2AC}}$, $\mathrm{Init}^2_{\mathrm{2AC}}$ and Rec in his simulation world generate the real world situation in which they exchange messages using $\mathrm{F_{DIC}}$. For $\mathrm{Sim_{2AC}}$ to establish the session, he inputs $in(\mathtt{Establish}_{\mathtt{2AC}},\mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}_i}$ and $in(\mathtt{Establish}_{\mathtt{2AC}},\mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Rec}}$ to $\mathrm{Init_{2AC_i}}$ and Rec, respectively. Finally, the parties establish two DICs in the simulation world. After the message is received by each party in the simulation world, they send $send(\mathtt{Establish}_{\mathtt{DIC}},\mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F^{(1,2)}_{DIC}}}$ to the $\mathrm{F_{DIC}}$. The states of each party becomes pre:$ntask$ = ESS2 from (L) in Table 7.10 and eff: $active := \top$ and $ntask := \bot$ from (K) and (L) in Table. 7.10. When $\mathrm{F_{DIC}}$ receives the messages, $active$ of the functionality becomes $\top$ from pre:(D), (E) and (G) in Table. 7.10. $\mathrm{F_{DIC}}$ then sends $send(\mathtt{SID},\mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ to the adversary in the simulation world (that may be $\mathrm{Sim_{2AC}}$). The state $ntask$ of $\mathrm{F_{DIC}}$ becomes $\bot$ from (I). If the adversary obtains the message, $active$ becomes $\top$ from (P) in Table. 7.11. The simulation in the establish session of the real world is perfectly executed by $\mathrm{Sim_{2AC}}$. Finally, the parties establish a DIC between $\mathrm{Init}^1_{\mathrm{2AC}}$ and $\mathrm{Init}^2_{\mathrm{2AC}}$ in the simulation world.

Simulation Policy

i. After receiving $receive(\mathtt{SID},\mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F_{2AC}}}$, $\mathrm{Sim_{2AC}}$ executes the following simulation.

A. $\mathrm{Sim_{2AC}}$ prepares the dummy parties, $\mathrm{Init_{2AC_i}}$, $\mathrm{Rec_{2AC}}$, and Adv, and ideal functionality task $\mathrm{F_{DIC}}$.

B. $\mathrm{Sim_{2AC}}$ inputs messages $in(\mathtt{Establish}_{\mathtt{DIC}},\mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init_{2AC_i}}}$ for each i and $in(\mathtt{Establish}_{\mathtt{DIC}},\mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec_{2AC}}}$ to

$\text{Init}_{2\text{AC}_\text{i}}$ and $\text{Rec}_{2\text{AC}}$.

C. $\text{Sim}_{2\text{AC}}$ makes $\text{Init}^1_{2\text{AC}}$ and $\text{Init}^2_{2\text{AC}}$ rach send message $\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\text{F}_{\text{DIC}}}$, respectively.

D. $\text{Sim}_{2\text{AC}}$ makes F_{DIC} send $\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\text{Adv}}$ to Adv.

Task Correspondence of Simulation

i. $\text{Init}^1_{2\text{AC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\text{F}_{\text{DIC}}}$
$=_{\text{corr.}}$ $\text{Sim}_{2\text{AC}}.\text{Init}^1_{2\text{AC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\text{F}_{\text{DIC}}}$

pre: $\mathit{ntask} = \text{ESS2}$; (L);

eff: $\mathit{active} := \top$ and $\mathit{ntask} := \perp$; $(K), (L)$;

ii. $\text{Init}^2_{2\text{AC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\text{F}_{\text{DIC}}}$
$=_{\text{corr.}}$ $\text{Sim}_{2\text{AC}}.\text{Init}^2_{2\text{AC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\text{F}_{\text{DIC}}}$

pre: $\mathit{ntask} = \text{ESS2}$; (L);

eff: $\mathit{active} := \top$ and $\mathit{ntask} := \perp$; $(K), (L)$;

iii. $\text{F}_{\text{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\text{Sim}_{2\text{AC}}}$
$=_{\text{corr.}}$ $\text{Sim}_{2\text{AC}}.\text{F}_{\text{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\text{Sim}_{2\text{AC}}}$

pre: $\mathit{ntask} = \text{ESS2}$; (I);

eff: $\mathit{ntask} := \perp$; (I);

The simulation of the establish session is perfectly executed by $\mathit{simulation}(*)$ of $\text{Sim}_{2\text{AC}}$. Finally, the parties establish a DIC in the simulation world.

(b) **Data Sending Session** Next, in the data sending session, Env sends message $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\overline{\text{Init}_i}}$ to $\overline{\text{Init}_{2\text{AC}_\text{i}}}$ for some i. Here, we consider that Env sends the message to $\overline{\text{Init}^1_{2\text{AC}}}$. The case of $\text{Init}^2_{2\text{AC}}$ is the same as that for $\text{Init}^1_{2\text{AC}}$. $\overline{\text{Init}_{2\text{AC}_\text{i}}}$ sends $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\text{F}_{2\text{AC}}}$ to $\text{F}_{2\text{AC}}$. $\text{F}_{2\text{AC}}$ then sends $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\text{Adv}}$ to $\text{Sim}_{2\text{AC}}$. After receiving the message, $\text{Sim}_{2\text{AC}}$ executes $\mathit{simulation}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})$ to mimic the data sending session in the real world. That is, he inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\text{Init}}$ to $\text{Init}^1_{2\text{AC}}$ in the simulation world. The parties, $\text{Init}^1_{2\text{AC}}$ and $\text{Init}^2_{2\text{AC}}$, exchange the message using a DIC. Moreover, $\text{Init}^1_{2\text{AC}}$ sends $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}^{(1,2)}_{\text{DIC}}}$ to F_{DIC}. F_{DIC} sends $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Adv}}$ to $\text{Sim}_{2\text{AC}}$. After receiving $\textit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{Adv}}$ from $\text{Sim}_{2\text{AC}}$, the functionality sends $\textit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, mes)_{\overline{X}}$ to the other sender $\text{Init}^2_{2\text{AC}}$. Here, $\text{Init}^1_{2\text{AC}}$ and $\text{Init}^2_{2\text{AC}}$ exchange the same sending message. Next, both senders send $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{Rec}}$ to receiver Rec. Rec receives message $\textit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{Init}_i}$ from both senders. Finally, Rec outputs $\textit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{Rec}}$. This is the basic simulation of $\text{Sim}_{2\text{AC}}$. He can simulate every case of the executions in the real world. After $\text{Sim}_{2\text{AC}}$ receives $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{Adv}}$, $\text{Sim}_{2\text{AC}}$ executes the task $\textit{simulation}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)$ in his simulation world under the policy described hereafter.

Simulation Policy

i. After receiving $\textit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{F}_{2\text{AC}}}$, $\text{Sim}_{2\text{AC}}$ executes the following simulation.

   A. $\text{Sim}_{2\text{AC}}$ executes $\textit{random}(*)$ and selects message input party, $\textit{party} \in \{\text{Init}^1_{2\text{AC}}, \text{Init}^2_{2\text{AC}}\}$. (The following discussion is $\textit{party} = \text{Init}^1_{2\text{AC}}$. The case of $\textit{party} = \text{Init}^2_{2\text{AC}}$ is analogous.)

   B. $\text{Sim}_{2\text{AC}}$ inputs $\textit{in}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}^1_{2\text{AC}}}$ to $\text{Init}^1_{2\text{AC}}$.

   C. $\text{Sim}_{2\text{AC}}$ makes $\text{Init}^1_{2\text{AC}}$ send $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$ to F_{DIC}.

   D. $\text{Sim}_{2\text{AC}}$ makes F_{DIC} receive $\textit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, mes)_{\text{Init}^1_{2\text{AC}}}$ and makes F_{DIC} send $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, mes)_{\text{Adv}}$.

   E. If F_{DIC} receives $\textit{send}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{F}_{\text{DIC}}}$ from Adv, $\text{Sim}_{2\text{AC}}$ continues the following.

   F. $\text{Sim}_{2\text{AC}}$ makes F_{DIC} receive $\textit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{Adv}}$ and makes F_{DIC} send $\textit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, mes)_{\text{Init}^2_{2\text{AC}}}$ to $\text{Init}^2_{2\text{AC}}$.

G. Sim_{2AC} makes Init^2_{2AC} receive $receive(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$, respectively.

H. Sim_{2AC} makes Init^1_{2AC} and Init^2_{2AC} each send $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{Rec}}$ according to $M_{\pi_{2AC}}$.

I. Sim_{2AC} makes Rec_{2AC} receive $receive(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}_{2AC_i}}$ and makes Rec_{2AC} output message $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Rec}_{2AC}}$.

ii. Sim_{2AC} executes $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, ok)_{\mathrm{F_{2AC}}}$.

The state changes and task correspondences are given hereafter.

Task Correspondence of Simulation

Init^1_{2AC} i. $(in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_i}$; pre: $active = \top$, mes and $ntask = \perp$:(J),(K),(L) ; eff:$mes := m$ and $ntask :=$ DSS2:(J), (L))

ii. $(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F^{(1,2)}_{DIC}}}$; pre: $s := mes$ and $ntask =$ DSS2: (L); eff: $ntask :=$ DSS4:(L))

$\mathrm{F_{DIC}}$ i. $(receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_X$; pre:$active = \top$, mes and $ntask = \perp$: (G), (H), (I); eff:$mes := m$ and $ntask :=$ DSS2:(H),(I))

ii. $(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$; pre:$okcond_{\mathrm{Adv}} = \perp$, $mes := m$ and $ntask =$ DSS2:(F), (H), (I); eff:$ntask :=$ DSS3: (I))

Adv i. $(receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$; pre: $active = \top$, $ntask = \perp$: (P), (Q); eff:$smes := m$, $ntask :=$ DSS2: (Q),(R))

ii. $(send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F_{DIC}}}$; pre: $ntask =$ DSS2: (Q) ; eff: $ntask := \perp$: (Q))

$\mathrm{F_{DIC}}$ i. $(receive(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{Adv}}$, pre:$ntask =$ DSS3: (I); eff:$okcond_{\mathrm{Adv}} := \top$ and $ntask :=$ DSS4:(F),(I))

ii. $(send(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, mes)_{\overline{X}}$; pre: $ntask =$ DSS4: (I); eff: $okcond_{\mathrm{Adv}}$, mes and $ntask := \perp$:(F),(H),(I))

Init^1_{2AC} i. $(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, s)_{\mathrm{Rec}}$; pre:$s := mes$ and $ntask =$ DSS4:(L) ; eff:mes and $ntask := \perp$:(J),(L))

$\text{Init}^2_{2\text{AC}}$ i. $(\textit{receive}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, r)_{\text{F}^{(1,2)}_{\text{DIC}}}$; pre: $\textit{active} = \top, \textit{mes}$ and $\textit{ntask} = \bot$:(J),(K) and (L) ; eff:$\textit{mes} := r$ and $\textit{ntask} :=$ DSS4 :(J),(L))

ii. $(\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, s)_{\text{Rec}}$; pre:$s := \textit{mes}$ and $\textit{ntask} =$ DSS4:(L) ; eff:$\textit{mes}$ and $\textit{ntask} := \bot$:(J),(L))

Rec i. $(\textit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{Init}_i}(i \in \{1,2\})$; pre: $\textit{active} = \top$ and $\textit{ntask} = \bot$:(O),(P) ;eff: If $\textit{mes} = \bot$, then $\textit{mes} := m$. Else, if $\textit{mes} = m$, then $\textit{ntask} :=$ DSS2 :(M),(O))

ii. $(\textit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{Rec}}$; pre:$m := \textit{mes}$ and $\textit{ntask} :=$ DSS2 :(M) and (O) ;eff:$\textit{mes}$ and $\textit{ntask} := \bot$:(M),(O))

Simulator $\text{Sim}_{2\text{AC}}$ executes the above-mentioned process to mimic the real world. The states correspondences in Table 7.10 and 7.11 work well. The key point of this simulation is as follows. To mimic the ideal world, the simulator executes the parties that execute the tasks in the real world. Moreover, not to distinguish the output trace, the simulator simulates the real world in his simulation world by using task codes. In the real world, $\text{Init}_{2\text{AC}_\text{i}}$ uses a DIC without the adversary being able to identify the direction of the sent message. Then $\text{Init}_{2\text{AC}_\text{i}}$ sends the shared message to $\text{Rec}_{2\text{AC}}$. In the simulation world, $\text{Sim}_{2\text{AC}}$ obtains the same output that $\text{Adv}_{2\text{AC}}$ outputs in the real world by his simulation. That is, the trace distributions of each world, the real world and the ideal world, are indistinguishable. In other words, since each task and state correspondences work well, the following property works well: $\textit{trace}(\epsilon'_\text{R}) = \textit{trace}(\epsilon'_\text{I})$.

(c) **Expire Session** Finally, in the expire session, Env sends message $\textit{in}(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\overline{\text{Init}_i}}$ and $\textit{in}(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\overline{\text{Rec}}}$ to $\overline{\text{Init}_{2\text{AC}_\text{i}}}$ and $\overline{\text{Rec}_{2\text{AC}}}$, respectively. $\overline{\text{Init}_{2\text{AC}_\text{i}}}$ relays message $\textit{send}(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\text{F}_{2\text{AC}}}$ to $\text{F}_{2\text{AC}}$. After receiving $\textit{receive}(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_X$ from $\text{F}_{2\text{AC}}$, $\text{Sim}_{2\text{AC}}$ terminates the session in the simulation world. That is, he inputs messages $\textit{in}(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\text{Init}_i}$ and $\textit{in}(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\text{Rec}}$

to Init_{2AC_i} and Rec_{2AC} in the simulation world.

Simulation Policy

i. After receiving $\mathit{receive}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F}_{2AC}}$, Sim_{2AC} executes the following simulation.

   A. Sim_{2AC} inputs messages $\mathit{in}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}^1_{2AC}}$ and $\mathit{in}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}^2_{2AC}}$ to Init^1_{2AC} and Init^2_{2AC}, respectively.

   B. Sim_{2AC} makes Init^1_{2AC} (resp., Init^2_{2AC}) send message $\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{DIC}}$ to F_{DIC}.

   C. Sim_{2AC} makes F_{DIC} send $\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ to Adv.

We assume that the state correspondences in Table 7.10 and 7.11 hold. From 3a and 3b, the state correspondences also hold after the simulation by Sim_{2AC}. That is, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

Task Correspondence of Simulation

i. $\mathrm{Init}^1_{2AC}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F}_{SC}}$
   $=_{\mathrm{corr.}}$ $\mathrm{Sim}_{2AC}.\mathrm{Init}^1_{2AC}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F}_{SC}}$

   pre: $\mathit{ntask} = \mathrm{EXS2}$; (L);

   eff: active and $\mathit{ntask} := \perp$; $(K), (L)$;

ii. $\mathrm{Init}^2_{2AC}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F}_{SC}}$
   $=_{\mathrm{corr.}}$ $\mathrm{Sim}_{2AC}.\mathrm{Init}^2_{2AC}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F}_{SC}}$

   pre: $\mathit{ntask} = \mathrm{EXS2}$; (L);

   eff: active and $\mathit{ntask} := \perp$; $(K), (L)$;

iii. $\mathrm{F}_{DIC}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}_{2AC}}$
   $=_{\mathrm{corr.}}$ $\mathrm{Sim}_{2AC}.\mathrm{F}_{DIC}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}_{2AC}}$

   pre: $\mathit{ntask} = \mathrm{EXS2}$; (I);

   eff: active, $\mathit{estcond}_X$ and $\mathit{ntask} := \perp$ for all X ; $(D) \sim (F)$, (I);

2. **Static Corruption**

This type of corruption is divided into the following three cases: only $\mathrm{Init}_{\mathrm{2AC_i}}$ is corrupted by Adv, only $\mathrm{Rec}_{\mathrm{2AC}}$ is corrupted by Adv, and both parties are corrupted by Adv.

(a) Only $\mathrm{Init}_{\mathrm{2AC_i}}$ for some or both i is corrupted by Adv This case means that $\mathrm{Adv}_{\mathrm{2AC}}$ corrupts only Init_i before the protocol starts. So, the remaining steps are identical to the above-mentioned No Corruption Case.

  i. After receiving the corrupt message from Env, $\mathrm{Sim}_{\mathrm{2AC}}$ prepares a situation in which only $\mathrm{Init}_{\mathrm{2AC_i}}$ for some i is corrupted and adds the following policy before 1(a)iB: $\mathrm{Sim}_{\mathrm{2AC}}$ makes Adv corrupt $\mathrm{Init}_{\mathrm{2AC_i}}$ for some i.

  ii. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{F_{2AC}}}$ from $\mathrm{F}_{\mathrm{2AC}}$, $\mathrm{Sim}_{\mathrm{2AC}}$ executes the following simulation.

    A. If message $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{F_{2AC}}}$ is input to corrupted party Init_i, $\mathrm{Sim}_{\mathrm{2AC}}$ inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{m}$ $\mathit{es})_{\mathrm{Init}_i}$ to Init_i.

    B. Else message $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{F_{2AC}}}$ is input to non-corrupted party$\mathrm{Init}_{\bar{i}}$, and $\mathrm{Sim}_{\mathrm{2AC}}$ inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{Init}_{\bar{i}}}$ to $\mathrm{Init}_{\bar{i}}$.

    C. The remaining asteps are the same as the simulation for No Corrupted Case.

  iii. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{F_{2AC}}}$ in $\overline{\mathrm{Init}_{\mathrm{2AC_i}}}$, $\mathrm{Sim}_{\mathrm{2AC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\overline{\mathrm{Init}_{\mathrm{2AC_i}}}}$.

$\mathrm{Adv}_{\mathrm{2AC}}$ and $\mathrm{Sim}_{\mathrm{2AC}}$ identify who sends the sending message, that is, Init_1 or Init_2. However, the simulation is perfectly executed. If the protocol executes the establish session, data sending session, and expire session, in any case, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by $\mathrm{Sim}_{\mathrm{2AC}}$. From the Task Correspondence in 7.2.2, the state correspondences in 7.9, 7.10, and 7.11 hold in this case. That is, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$ holds.

(b) Only $\text{Rec}_{2\text{AC}}$ is corrupted by Adv

This case means that $\text{Adv}_{2\text{AC}}$ corrupts only $\text{Rec}_{2\text{AC}}$ before the protocol starts. $\text{Adv}_{2\text{AC}}$ and $\text{Sim}_{2\text{AC}}$ do not identify who sends the sending message.

i. After receiving the corrupt message from Env, $\text{Sim}_{2\text{AC}}$ prepares a situation in which only $\text{Rec}_{2\text{AC}}$ is corrupted and adds the following policy before 1(a)iB: $\text{Sim}_{2\text{AC}}$ makes Adv corrupt $\text{Rec}_{2\text{AC}}$.

ii. The remaining simulation is the same as the simulation for the No Corrupted Case.

iii. After receiving $\textit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{F}_{2\text{AC}}}$ in $\overline{\text{Rec}_{2\text{AC}}}$, $\text{Sim}_{2\text{AC}}$ executes $\textit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, m)_{\overline{\text{Rec}_{2\text{AC}}}}$.

Additionally, $\text{Rec}_{2\text{AC}}$ has no secret information. So, the simulation after the corruption is perfectly executed. If the protocol executes the establish session, data sending session, and expire session, in any case, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by $\text{Sim}_{2\text{AC}}$. From the Task Correspondence in 7.2.2, the state correspondences in 7.9, 7.10, and 7.11 hold in this case.

(c) Both parties are corrupted by Adv

This case means that $\text{Adv}_{2\text{AC}}$ corrupts both $\text{Init}_{2\text{AC}_\text{i}}$ for both i ($\text{Init}^1_{2\text{AC}}$ and $\text{Init}^2_{2\text{AC}}$) and $\text{Rec}_{2\text{AC}}$ before the protocol starts.

i. After receiving the corrupt message from Env, Sim_{DIC} prepares a situation in which only Init and Rec are corrupted adding the following policy before 1(a)iB: $\text{Sim}_{2\text{AC}}$ makes Adv corrupt $\text{Init}^1_{2\text{AC}}$, $\text{Init}^2_{2\text{AC}}$, and $\text{Rec}_{2\text{AC}}$.

ii. If the data sending message is input to $\textit{party} \in \{\text{Init}, \text{Rec}\}$, $\text{Sim}_{2\text{AC}}$ inputs $\textit{in}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\textit{party}}$ to $\textit{party}$.

iii. The remaining is the same as the simulation for the No Corrupted Case.

iv. After receiving $receive(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{F}_{\mathrm{2AC}}}$ in $\overline{party}$ in his simulation, $\mathrm{Sim}_{\mathrm{2AC}}$ executes $out(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, m)_{\overline{party}}$.

In this case, $\mathrm{Adv}_{\mathrm{2AC}}$ and $\mathrm{Sim}_{\mathrm{2AC}}$ can identify who sends the sending message. However, the simulation is perfectly executed. If the protocol executes the establish session, data sending session, and expire session, in any case, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by $\mathrm{Sim}_{\mathrm{2AC}}$. From the Task Correspondence in 7.2.2, the state correspondences in 7.9, 7.10, and 7.11 hold in this case. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$ holds.

3. **Adaptive Corruption**

In this case, the adversary corrupts some parties when he wants to do so. This case is also simulated by the simulator, but the direction that the message is sent cannot be concealed from the adversary after he corrupts some parties. However, this case is also simulated by simulator $\mathrm{Sim}_{\mathrm{2AC}}$, so the simulation is perfectly executed. This case is separated into the following instances.

(a) **Establish Session**

Instance 1: Before $\mathrm{Init}_{\mathrm{2AC_i}}$ and $\mathrm{Rec}_{\mathrm{2AC}}$ are activated.

Instance 2: After $\mathrm{Init}^1_{\mathrm{2AC}}$ is activated but before $\mathrm{Init}^2_{\mathrm{2AC}}$ and $\mathrm{Rec}_{\mathrm{2AC}}$ are activated.

Instance 3: After $\mathrm{Init}^2_{\mathrm{2AC}}$ activated and before $\mathrm{Init}^1_{\mathrm{2AC}}$ and $\mathrm{Rec}_{\mathrm{2AC}}$ are activated.

Instance 4: After $\mathrm{Rec}_{\mathrm{2AC}}$ is activated but before $\mathrm{Init}_{\mathrm{2AC_i}}$ is activated

Instance 5: After $\mathrm{Init}_{\mathrm{2AC_i}}$ is activated but before $\mathrm{Rec}_{\mathrm{2AC}}$ is activates.

Instance 6: After $\mathrm{Init}^1_{\mathrm{2AC}}$ and $\mathrm{Rec}_{\mathrm{2AC}}$ are activated but before $\mathrm{Init}^2_{\mathrm{2AC}}$ is activated.

Instance 7: After $\mathrm{Init}^2_{\mathrm{2AC}}$ and $\mathrm{Rec}_{\mathrm{2AC}}$ are activated but before $\mathrm{Init}^1_{\mathrm{2AC}}$ is activates.

Instance 8: After Init_{2AC_i} and Rec_{2AC} are activated.

These case are analogous to case 2 because there is no secret information. The adversary can corrupt Init^1_{2AC}, Init^2_{2AC} or Rec_{2AC}, or any combination, but the simulator can also corrupt the corresponding parties. These cases are also perfectly simulated by Sim_{2AC}.

(b) **Data Sending Session**

Instance 1: Before or after Init^1_{2AC} or Rec_{2AC} is activated by receiving $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_i}$ from the Env.
Env can execute only the message sending indication and the corrupt indication. So, this case is only the case that the adversary corrupts the party. This case is also simulated by Sim_{2AC}, because there is no secret information in this session. The task correspondence works well and there exists a simulation relation between the real world and ideal world. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$ holds.

(c) **Expire Session**

Instance 1: After Init^1_{2AC}, Init^2_{2AC}, or Rec_{2AC} is activated with the expire message.
Once Env sends the expire message to Init_{2AC_i} or Rec_{2AC}, this session terminates in the real and ideal world. So the adversary can corrupt the parties. This is identical to case 2.

Simulation Policy

Sim_{2AC} simulates in his simulation world the following.

(a) After receiving "corrupt Init^1_{2AC}" message from Env,

- Sim_{2AC} corrupts $\overline{\mathrm{Init}^1_{2AC}}$ and checks whether $party \in \{\mathrm{Init}^1_{2AC}, \mathrm{Init}^2_{2AC}\}$ has already sent the data sending message to the other party. If the message was already sent, Sim_{2AC}

performs the following. Else, $\mathrm{Sim}_{\mathrm{2AC}}$ makes Adv corrupt $\mathrm{Init}^1_{\mathrm{2AC}}$.

- If *party* = $\mathrm{Init}^1_{\mathrm{2AC}}$,
  - If $\mathrm{Sim}_{\mathrm{2AC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^1_{\mathrm{2AC}}}$ to $\mathrm{Init}^1_{\mathrm{2AC}}$ in his simulation, then $\mathrm{Sim}_{\mathrm{2AC}}$ simulates that Adv corrupts $\mathrm{Init}^1_{\mathrm{2AC}}$, immediately.
  - Else, $\mathrm{Sim}_{\mathrm{2AC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^2_{\mathrm{2AC}}}$ to $\mathrm{Init}^2_{\mathrm{2AC}}$ in his simulation, then $\mathrm{Sim}_{\mathrm{2AC}}$ simulates that Adv corrupts $\mathrm{Init}^2_{\mathrm{2AC}}$, immediately.
- Else, *party* = $\mathrm{Init}^2_{\mathrm{2AC}}$,
  - If $\mathrm{Sim}_{\mathrm{2AC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^1_{\mathrm{2AC}}}$ to $\mathrm{Init}^1_{\mathrm{2AC}}$ in his simulation, then $\mathrm{Sim}_{\mathrm{2AC}}$ simulates that Adv corrupts $\mathrm{Init}^2_{\mathrm{2AC}}$, immediately.
  - Else, $\mathrm{Sim}_{\mathrm{2AC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^2_{\mathrm{2AC}}}$ to $\mathrm{Init}^2_{\mathrm{2AC}}$ in his simulation, then $\mathrm{Sim}_{\mathrm{2AC}}$ simulates that Adv corrupts $\mathrm{Init}^1_{\mathrm{2AC}}$, immediately.
- If more data sending messages are input to *party* from Env after $\mathrm{Sim}_{\mathrm{2AC}}$ corrupts *party*, $\mathrm{Sim}_{\mathrm{2AC}}$ can also simulate the situation. If the message is input to corrupted $\mathrm{Init}^1_{\mathrm{2AC}}$, $\mathrm{Sim}_{\mathrm{2AC}}$ inputs the sending message to the corrupted *party* in his simulation. Else, the message is input to $\mathrm{Init}^2_{\mathrm{2AC}}$ and $\mathrm{Sim}_{\mathrm{2AC}}$ inputs the sending message to non-corrupted *party* in his simulation.
- After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{F}_{\mathrm{2AC}}}$ in $\overline{\mathrm{Init}^1_{\mathrm{2AC}}}$, $\mathrm{Sim}_{\mathrm{2AC}}$ executes $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\overline{\mathrm{Init}^1_{\mathrm{2AC}}}}$.

(b) After receiving "corrupt $\mathrm{Init}^2_{\mathrm{2AC}}$" message from Env,

- Sim_{2AC} corrupts $\overline{\mathrm{Init}^2_{2AC}}$ and checks whether *party* $\in \{\mathrm{Init}^1_{2AC}, \mathrm{Init}^2_{2AC}\}$ has already sent the data sending message to the other party. If the message was already sent, Sim_{2AC} performs the following. Else, Sim_{2AC} makes Adv corrupt Init^2_{2AC}.
- If *party* $= \mathrm{Init}^1_{2AC}$,
  - If Sim_{2AC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^1_{2AC}}$ to Init^1_{2AC} in his simulation, then Sim_{2AC} simulates that Adv corrupts Init^2_{2AC}, immediately.
  - Else, Sim_{2AC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^2_{2AC}}$ to Init^2_{2AC} in his simulation, then Sim_{2AC} simulates that Adv corrupts Init^1_{2AC}, immediately.
- Else, *party* $= \mathrm{Init}^2_{2AC}$,
  - If Sim_{2AC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^1_{2AC}}$ to Init^1_{2AC} in his simulation, then Sim_{2AC} simulates that Adv corrupts Init^1_{2AC}, immediately.
  - Else, Sim_{2AC} has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}^2_{2AC}}$ to Init^2_{2AC} in his simulation, then Sim_{2AC} simulates that Adv corrupts Init^2_{2AC}, immediately.
- If more data sending messages are input to *party* from Env after Sim_{2AC} corrupted *party*, Sim_{2AC} can also simulate the situation. If the message is input to corrupted Init^2_{2AC}, Sim_{2AC} inputs the sending message to the corrupted *party* in his simulation. Else, if the message is input to Init^1_{2AC}, Sim_{2AC} inputs the sending message to non-corrupted *party* in his simulation.
- Receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{F}_{2AC}}$ in $\overline{\mathrm{Init}^2_{2AC}}$, Sim_{2AC}

executes $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\overline{\mathrm{Init}^2_{\mathrm{2AC}}}}$.

(c) After receiving “corrupt $\mathrm{Rec}_{\mathrm{2AC}}$” message from Env,

- $\mathrm{Sim}_{\mathrm{2AC}}$ corrupts $\overline{\mathrm{Rec}_{\mathrm{2AC}}}$ and makes Adv corrupt $\mathrm{Rec}_{\mathrm{2AC}}$ in the simulation world, immediately.
- After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{F}_{\mathrm{2AC}}}$ in $\overline{\mathrm{Rec}_{\mathrm{2AC}}}$, $\mathrm{Sim}_{\mathrm{2AC}}$ executes $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\overline{\mathrm{Rec}_{\mathrm{2AC}}}}$.

(d) After receiving “corrupt $\mathrm{Init}^1_{\mathrm{2AC}}$ and $\mathrm{Init}^2_{\mathrm{2AC}}$” message from Env,

- $\mathrm{Sim}_{\mathrm{2AC}}$ corrupts $\overline{\mathrm{Init}^1_{\mathrm{2AC}}}$ and $\overline{\mathrm{Init}^2_{\mathrm{2AC}}}$ and checks which $party \in \{\mathrm{Init}_1, \mathrm{Init}_2\}$ sent the message to the other party. If $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{party}$ was already sent, $\mathrm{Sim}_{\mathrm{2AC}}$ makes Adv corrupt Init and Rec and does the following. Else, $\mathrm{Sim}_{\mathrm{2AC}}$ makes Adv corrupt Init and Rec.
  - If the party that $\mathrm{Sim}_{\mathrm{2AC}}$ has already input the message sending request is equal to the party to which Env input a message, $\mathrm{Sim}_{\mathrm{2AC}}$ inputs more data sending requests to the party.
  - Else, the input party in the simulation world is not same as the input party in the ideal world, $\mathrm{Sim}_{\mathrm{2AC}}$ regards the input party in the simulation world as the input party which has already input a message in the ideal world. The other party in the simulation world is also regarded as the party which has not input a message yet in the ideal world.
- After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{F}_{\mathrm{2AC}}}$ in $\overline{\mathrm{Init}_{\mathrm{2AC}_i}}$, $\mathrm{Sim}_{\mathrm{2AC}}$ executes $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\overline{\mathrm{Init}_{\mathrm{2AC}_i}}}$.

(e) After receiving “corrupt $\mathrm{Init}^1_{\mathrm{2AC}}$ and $\mathrm{Rec}_{\mathrm{2AC}}$” message from Env,

- $\mathrm{Sim}_{\mathrm{2AC}}$ corrupts $\overline{\mathrm{Init}^1_{\mathrm{2AC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{2AC}}}$, and makes Adv corrupt $\mathrm{Rec}_{\mathrm{2AC}}$ and Init_i according to case 3a.

(f) After receiving "corrupt Init^2_{2AC} and Rec_{2AC}" message from Env,

- Sim_{2AC} corrupts $\overline{\mathrm{Init}^2_{2AC}}$ and $\overline{\mathrm{Rec}_{2AC}}$, and makes Adv corrupt Rec_{2AC} and Init_i according to case 3b.

(g) After receiving "corrupt Init^1_{2AC}, Init^2_{2AC} and Rec_{2AC}" message from Env,

- Sim_{2AC} corrupts $\overline{\mathrm{Init}^1_{2AC}}$, $\overline{\mathrm{Init}^2_{2AC}}$ and $\overline{\mathrm{Rec}_{2AC}}$, and makes Adv corrupt Rec_{2AC} and Init_i for each i according to case 3d.

Whenever Adv_{2AC} corrupts some party, Sim_{2AC} corrupts the corresponding dummy party in the ideal world and forwards the obtained information to the simulated copy of Adv_{2AC}. If Adv_{2AC} corrupts party Init_{2AC_i} or Rec_{2AC} then Sim_{2AC} corrupts $\overline{\mathrm{Init}_{DIC}}$ or (and) $\overline{\mathrm{Rec}_{DIC}}$ in the ideal world, and provides the simulated copy of Adv_{2AC} in the simulation world with the state information of the corrupted party. Conversely, Sim_{DIC} may obtain information from the simulated world with the corruptions. Additionally, in this protocol party has no secret information because F_{DIC} is securely performed. In all cases, since Sim_{2AC} can simulate Adv_{2AC} using his simulated world, Env cannot distinguish the real world from the ideal world. That is, simulating party corruption is perfectly executed.

Finally, relation R is a simulation relation based on the task and state correspondences with respect to the adaptive adversary. We obtain Lemma2.

□

Next, Theorem5 is obtained from Lemma2 immediately.

Proof. From Lemma2 and Theorem3, Theorem5 is proved. That is, the trace distribution property, $tdist(\epsilon_R) = tdist(\epsilon_I)$ holds with respect to adaptive adversary under $M_{\pi_{2AC}}$. As a result, the simulation is perfectly executed because Sim_{2AC} can simulate the real world from the information message through $\mathrm{Adv}_{\mathrm{2AC}}$. The tasks of the real world perfectly correspond to the the tasks in the ideal world. That is,

$$\mathrm{Real}_{\mathrm{2AC}} \| \mathrm{Env} \;\; \mathrm{Hyb.} \leq_0^{\mathtt{M}_{\pi_{\mathrm{2AC}}}} \mathrm{Ideal}_{\mathrm{2AC}} \| \mathrm{Env}.$$

□

Functionality

(a)	$u.\mathrm{F}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Init}_1} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Init}}$
(b)	$u.\mathrm{F}_{\mathrm{2AC}}.\mathit{estcond}_{\mathrm{Init}_2} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Rec}}$
(c)	$u.\mathrm{F}_{\mathrm{2AC}}.\mathit{okcond}_{\mathrm{Adv}} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{okcond}_{\mathrm{Adv}}$
(d)	$u.\mathrm{F}_{\mathrm{2AC}}.\mathit{active} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{active}$
(e)	$u.\mathrm{F}_{\mathrm{2AC}}.\mathit{mes} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{mes}$
(f)	$u.\mathrm{F}_{\mathrm{2AC}}.\mathit{ntask} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{ntask}$

Initiator

(g)	$u.\overline{\mathrm{Init}_{\mathrm{2AC}_\mathrm{i}}}.\mathit{mes} = s.\mathrm{Init}_{\mathrm{2AC}_\mathrm{i}}.\mathit{mes}$
(h)	$u.\overline{\mathrm{Init}_{\mathrm{2AC}_\mathrm{i}}}.\mathit{active} = s.\mathrm{Init}_{\mathrm{2AC}_\mathrm{i}}.\mathit{active}$
(i)	$u.\overline{\mathrm{Init}_{\mathrm{2AC}_\mathrm{i}}}.\mathit{ntask} = s.\mathrm{Init}_{\mathrm{2AC}_\mathrm{i}}.\mathit{ntask}$

Receiver

(j)	$u.\overline{\mathrm{Rec}_{\mathrm{2AC}}}.\mathit{mes} = s.\mathrm{Rec}_{\mathrm{2AC}}.\mathit{mes}$
(k)	$u.\overline{\mathrm{Rec}_{\mathrm{2AC}}}.\mathit{active} = s.\mathrm{Rec}_{\mathrm{2AC}}.\mathit{active}$
(l)	$u.\overline{\mathrm{Rec}_{\mathrm{2AC}}}.\mathit{ntask} = s.\mathrm{Rec}_{\mathrm{2AC}}.\mathit{ntask}$

Environment

(m)	$u.\mathrm{Env} = s.\mathrm{Env}$

Table 7.9: State Correspondence for $\mathrm{Real}_{\mathrm{2AC}}$ and $\mathrm{Ideal}_{\mathrm{2AC}}$ (Part I)

Simulator (or Adversary)

(A)	$u.\mathrm{Sim}_{2AC}.active = s.\mathrm{Adv}_{2AC}.active$
(B)	$u.\mathrm{Sim}_{2AC}.ntask = s.\mathrm{Adv}_{2AC}.ntask$
(C)	$u.\mathrm{Sim}_{2AC}.smes = s.\mathrm{Adv}_{2AC}.smes$
(D)	$u.\mathrm{Sim}_{2AC}.\mathrm{F}_{DIC}.estcond_{\mathrm{Init}_1} = s.\mathrm{F}_{DIC}.estcond_{\mathrm{Init}_1}$
(E)	$u.\mathrm{Sim}_{2AC}.\mathrm{F}_{DIC}.estcond_{\mathrm{Init}_2} = s.\mathrm{F}_{DIC}.estcond_{\mathrm{Init}_2}$
(F)	$u.\mathrm{Sim}_{2AC}.\mathrm{F}_{DIC}.okcond_{\mathrm{Adv}} = s.\mathrm{F}_{DIC}.okcond_{\mathrm{Adv}}$
(G)	$u.\mathrm{Sim}_{2AC}.\mathrm{F}_{DIC}.active = s.\mathrm{F}_{DIC}.active$
(H)	$u.\mathrm{Sim}_{2AC}.\mathrm{F}_{DIC}.mes = s.\mathrm{F}_{DIC}.mes$
(I)	$u.\mathrm{Sim}_{2AC}.\mathrm{F}_{DIC}.ntask = s.\mathrm{F}_{DIC}.ntask$
(J)	$u.\mathrm{Sim}_{2AC}.\mathrm{Init}_{2AC_i}.mes = s.\mathrm{Init}_{2AC_i}.mes$
(K)	$u.\mathrm{Sim}_{2AC}.\mathrm{Init}_{2AC_i}.active = s.\mathrm{Init}_{2AC_i}.active$
(L)	$u.\mathrm{Sim}_{2AC}.\mathrm{Init}_{2AC_i}.ntask = s.\mathrm{Init}_{2AC_i}.ntask$

Table 7.10: State Correspondence for Real_{2AC} and Ideal_{2AC} (Part II)

Simulator (or Adversary)

(M)	$u.\mathrm{Sim}_{2AC}.\mathrm{Rec}_{2AC}.mes = s.\mathrm{Rec}_{2AC}.mes$
(N)	$u.\mathrm{Sim}_{2AC}.\mathrm{Rec}_{2AC}.active = s.\mathrm{Rec}_{2AC}.active$
(O)	$u.\mathrm{Sim}_{2AC}.\mathrm{Rec}_{2AC}.ntask = s.\mathrm{Rec}_{2AC}.ntask$
(P)	$u.\mathrm{Sim}_{2AC}.\mathrm{Adv}_{2AC}.active = s.\mathrm{Adv}_{2AC}.active$
(Q)	$u.\mathrm{Sim}_{2AC}.\mathrm{Adv}_{2AC}.ntask = s.\mathrm{Adv}_{2AC}.ntask$
(R)	$u.\mathrm{Sim}_{2AC}.\mathrm{Adv}_{2AC}.smes = s.\mathrm{Adv}_{2AC}.smes$

Table 7.11: State Correspondence for Real_{2AC} and Ideal_{2AC} (Part III)

1. **Establish Session**

(a)	$\mathrm{Init}_{2AC_i}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{F_{DIC}^{(1,2)}}$ $=_{\text{corr.}} \overline{\mathrm{Init}_{2AC_i}}.\mathit{send}(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{F_{2AC}}$
(b)	$F_{DIC}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{Adv} =_{\text{corr.}} F_{2AC}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{Adv}$

2. **Expire Session**

(a)	$\mathrm{Init}_{2AC_i}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{F_{DIC}^{(1,2)}}$ $=_{\text{corr.}} \overline{\mathrm{Init}_{2AC_i}}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{F_{2AC}}$
(b)	$F_{DIC}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{Adv} =_{\text{corr.}} F_{2AC}.\mathit{send}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{Adv}$

3. **Environment**

(a)	All tasks of environment Env in Real_{2AC} correspond to the tasks of environment in Ideal_{2AC}.

Table 7.12: Corresponding Tasks for Real_{2AC} and Ideal_{2AC}

No.	**Real Execution**
1	$\mathrm{Init}^1_{\mathrm{2AC}}.\mathit{in}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{Init}_i}$
2	$\mathrm{Init}^1_{\mathrm{2AC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
3	$\mathrm{F}_{\mathrm{DIC}}.\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_X$
4	$\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{Adv}}$
5	$\mathrm{Adv}_{\mathrm{2AC}}.\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
6	$\mathrm{Adv}_{\mathrm{2AC}}.\mathit{send}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{DIC}}}$
7	$\mathrm{F}_{\mathrm{DIC}}.\mathit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, \mathit{ok})_{\mathrm{Adv}}$
8	$\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, \mathit{mes})_{\overline{X}}$
9	$\mathrm{Init}^2_{\mathrm{2AC}}.\mathit{receive}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
10	$\{\mathrm{Init}^1_{\mathrm{2AC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\mathrm{Rec}}, \mathrm{Init}^2_{\mathrm{2AC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\mathrm{Rec}}\}$
11	$\mathrm{Rec}_{\mathrm{2AC}}.\mathit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{Init}_i}$
12	$\mathrm{Rec}_{\mathrm{2AC}}.\mathit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{Init}_i}$
13	$\mathrm{Rec}_{\mathrm{2AC}}.\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{Rec}}$

No.	**Ideal Execution**
1	$\overline{\underline{\mathrm{Init}^1_{\mathrm{2AC}}}}.\mathit{in}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\overline{\mathrm{Init}_i}}$
2	$\mathrm{Init}^1_{\mathrm{2AC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{F}_{\mathrm{2AC}}}$
3	$\mathrm{F}_{\mathrm{2AC}}.\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\mathrm{Init}_i}$
4	$\mathrm{F}_{\mathrm{2AC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, \mathit{mes})_{\mathrm{Adv}}$
5	$\mathrm{Sim}_{\mathrm{2AC}}.\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, \mathit{mes})_{\mathrm{F}_{\mathrm{2AC}}}$
6	$\mathrm{Sim}_{\mathrm{2AC}}.\mathit{send}(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{2AC}}}$
7	$\mathrm{F}_{\mathrm{2AC}}.\mathit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, \mathit{ok})_{\mathrm{Adv}}$
8	$\mathrm{F}_{\mathrm{2AC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, \mathit{mes})_{\mathrm{Rec}}$
9	$\mathrm{Rec}_{\mathrm{2AC}}.\mathit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, \mathit{mes})_{\mathrm{F}_{\mathrm{2AC}}}$
10	$\mathrm{Rec}_{\mathrm{2AC}}.\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, m)_{\overline{\mathrm{Rec}}}$

Table 7.13: Corresponding Task Sequence of Data Sending Session for $\mathrm{Real}_{\mathrm{2AC}}$ and $\mathrm{Ideal}_{\mathrm{2AC}}$ under $M_{\pi_{\mathrm{2AC}}}$

Real Execution

	Init^1_{2AC}						Init^2_{2AC}					
	active		*ntask*		*mes*		*active*		*ntask*		*mes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	m	$\top$	-	$\perp$	DSS2	$\perp$	m
DSS2	-	-	DSS2	DSS4	-	-	-	-	DSS2	DSS4	-	-
DSS3	$\top$	-	$\perp$	DSS4	$\perp$	r	$\top$	-	$\perp$	DSS4	$\perp$	r
DSS4	-	-	DSS4	$\perp$	-	$\perp$	-	-	DSS4	$\perp$	-	$\perp$
Initial value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	

Ideal Execution

	$\overline{\mathrm{Init}^1_{2AC}}$						$\overline{\mathrm{Init}^2_{2AC}}$					
	active		*ntask*		*mes*		*active*		*ntask*		*mes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	m	$\top$	-	$\perp$	DSS2	$\perp$	m
DSS2	-	-	DSS2	$\perp$	-	$\perp$	-	-	DSS2	$\perp$	-	$\perp$
DSS3	-	-	-	-	-	-	-	-	-	-	-	-
DSS4	-	-	-	-	-	-	-	-	-	-	-	-
Initial value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	

Table 7.14: State Flow of Data Sending Session for Real_{2AC} and Ideal_{2AC} (Part I)

Real Execution

	Rec_{2AC}						F_{DIC}							
	active		*ntask*		*mes*		*active*		*ntask*		*mes*		$okcond_{Adv}$	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	*	$\perp$	m	$\top$	-	$\perp$	DSS2	$\perp$	m	-	-
DSS2	-	-	DSS2	$\perp$	-	$\perp$	-	-	DSS2	DSS3	-	-	$\perp$	-
DSS3	-	-	-	-	-	-	-	-	DSS3	DSS4	-	-	-	$\top$
DSS4	-	-	-	-	-	-	-	-	DSS4	$\perp$	-	$\perp$	-	$\perp$
Initial value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	

* If $mes = \perp$ then $mes := m$. Else if $mes = m$ then $ntask :=$ DSS2. Else $mes := \perp$.

Ideal Execution

	$\overline{\mathrm{Rec}_{2AC}}$						F_{2AC}							
	active		*ntask*		*mes*		*active*		*ntask*		*mes*		$okcond_{Adv}$	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	m	$\top$	-	$\perp$	DSS2	$\perp$	m	-	-
DSS2	-	-	DSS2	$\perp$	-	$\perp$	-	-	DSS2	DSS3	-	-	$\perp$	-
DSS3	-	-	-	-	-	-	-	-	DSS3	DSS4	-	-	-	$\top$
DSS4	-	-	-	-	-	-	-	-	DSS4	$\perp$	-	$\perp$	-	$\perp$
Initial value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$		$\perp$	

Table 7.15: State Flow of Data Sending Session for Real_{2AC} and Ideal_{2AC} (Part II)

Real Execution

Adv_{2AC}

	active		*ntask*		*smes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	-	m
DSS2	-	-	DSS2	$\perp$	-	-
DSS3	-	-	-	-	-	-
DSS4	-	-	-	-	-	-
Initial value	$\top$		$\perp$		-	
Final value	$\top$		$\perp$		m	

Ideal Execution

Sim_{2AC}

	active		*ntask*		*smes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	-	m
DSS2	-	-	DSS2	$\perp$	-	-
DSS3	-	-	-	-	-	-
DSS4	-	-	-	-	-	-
Initial value	$\top$		$\perp$		-	
Final value	$\top$		$\perp$		m	

Table 7.16: State Flow of Data Sending Session for Real_{2AC} and Ideal_{2AC} (Part III)

Code for Initiator $\mathrm{Init}_i (i \in \{1,2\})$ of Two Anonymous Channel, Init_{2AC_i}

Signature:

$\mathtt{sid}_{\mathtt{2AC}} = (\{\mathrm{Init}_1, \mathrm{Init}_2\}, \mathrm{Rec}, \mathtt{sid}'_{\mathtt{2AC}})$
$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}_1, \mathrm{Init}_2\}, \mathtt{sid}_{\mathtt{DIC}}')$

Input:	Output:
$in(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}_i}$	$send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
$in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_i}$	$send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
$receive(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$	$send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{Rec}}$
$in(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}_i}$	$send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$

State:

$active \in \{\perp, \top\}$, initially $\perp$ $\qquad$ $mes \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$

Tasks:

$\{send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}},$
$send(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, s)_{\mathrm{Rec}}, send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}\}$

Figure 7.23: Code for Initiator of Two Anonymous Channel, Init_{2AC} (Part I)

```
Code for Initiator $\mathrm{Init}_i (i \in \{1,2\})$ of Two Anonymous Channel, $\mathrm{Init}_{2AC_i}$

Transitions:

    Establish Session:

        ESS1. $in(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}_i}$
                pre: $active$ and $ntask = \perp$
                eff: $ntask := \mathrm{ESS2}$
        ESS2. $send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
                pre: $ntask = \mathrm{ESS2}$
                eff: $active := \top$ and $ntask := \perp$

    Data Sending Session:

        DSS1. $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_i}$
                pre: $active = \top$, $mes$ and $ntask = \perp$
                eff: $mes := m$ and $ntask := \mathrm{DSS2}$
        DSS2. $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
                pre: $s := mes$ and $ntask = \mathrm{DSS2}$
                eff: $ntask := \mathrm{DSS4}$
        DSS3. $receive(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, r)_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
                pre: $active = \top, mes$ and $ntask = \perp$
                eff: $mes := r$ and $ntask := \mathrm{DSS4}$
        DSS4. $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, s)_{\mathrm{Rec}}$
                pre: $s := mes$ and $ntask = \mathrm{DSS4}$
                eff: $mes$ and $ntask := \perp$

    Expire Session:

        EXS1. $in(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Init}_i}$
                pre: $active = \top$ and $ntask = \perp$
                eff: $ntask := \mathrm{EXS2}$
        EXS2. $send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}^{(1,2)}_{\mathrm{DIC}}}$
                pre: $ntask = \mathrm{EXS2}$
                eff: $active := \perp$ and $ntask := \perp$
```

Figure 7.24: Code for Initiator of Two Anonymous Channel, Init_{2AC} (Part II)

Code for Receiver Rec of Two Anonymous Channel, Rec_{2AC}

Signature:

$\texttt{sid}_{2AC} = (\{\text{Init}_1, \text{Init}_2\}, \text{Rec}, \texttt{sid}'_{2AC})$

Input:	Output:
$in(\texttt{Establish}_{2AC}, \texttt{sid}_{2AC})_{\text{Rec}}$	
$receive(\texttt{Response}, \texttt{sid}_{2AC}, m)_{\text{Init}_i}$	$out(\texttt{Receive}, \texttt{sid}_{2AC}, m)_{\text{Rec}}$
$in(\texttt{Expire}_{2AC}, \texttt{sid}_{2AC})_{\text{Rec}}$	

State:

$active \in \{\perp, \top\}$, initially $\perp$	$mes \in (\{0,1\}^*) \cup \{\perp\}$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$	

Tasks:

$\{out(\texttt{Receive}, \texttt{sid}_{2AC}, m)_{\text{Rec}}\}$

Figure 7.25: Code for Receiver of Two Anonymous Channel, Rec_{2AC} (Part I)

```
Code for Receiver Rec of Two Anonymous Channel, $\mathrm{Rec}_{2AC}$

Transitions:

    Establish Session:

        ESS1. $in(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Rec}}$
            pre: $active$ and $ntask = \perp$
            eff: $active := \top$

    Data Sending Session:

        DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Init}_i} (i \in \{1, 2\})$
            pre: $active = \top$ and $ntask = \perp$
            eff: If $mes = \perp$ then $mes := m$.
            Else if $mes = m$ then $ntask :=$ DSS2.
            Else $mes := \perp$
        DSS2. $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, m)_{\mathrm{Rec}}$
            pre: $m := mes$ and $ntask :=$ DSS2
            eff: $mes$ and $ntask := \perp$

    Expire Session:

        EXS1. $in(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Rec}}$
            pre: $active = \top$, $mes$ and $ntask = \perp$
            eff: $active$ and $ntask := \perp$
```

Figure 7.26: Code for Receiver of Two Anonymous Channel, Rec_{2AC} (Part II)

Code for Adversary for Annonymous Channel, $\mathrm{Adv}_{2\mathrm{AC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathtt{DIC}})$

Input:
$\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{receive}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

Output:
$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$

Other:
*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message, $\mathit{send}(m)_{\mathit{party}}, \mathit{receive}(m)_{\mathit{party}}, \mathit{out}(*)$, where $\mathit{party} \in \{\mathrm{Init}_1, \mathrm{Init}_2, \mathrm{Rec}\}$.

State:

$\mathit{active} \in \{\perp, \top\}$, initially $\perp$ $\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$\mathit{smes} \in (\{0,1\}) \cup \{\perp\}$, initially $\perp$

Tasks:

$\{\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}\}$

Figure 7.27: Code for Adversary for two Annonymous Channel, $\mathrm{Adv}_{2\mathrm{AC}}$ (Part I)

Code for Adversary for Annonymous Channel, Adv_{2AC}

```
Transitions:

    Establish Session:
        ESS1. $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{F_{DIC}}$
            pre: $active = \perp$
            eff:$active := \top$
    Data Sending Session:
        DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{F_{DIC}}$
            pre: $active = \top$, $ntask = \perp$
            eff: $smes := m$, $ntask :=$ DSS2
        DSS2. $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{F_{DIC}}$
            pre: $ntask =$ DSS2
            eff: $ntask := \perp$
    Expire Session:
        EXS1. $receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{F_{DIC}}$
            pre: $active = \top$
            eff: $active := \perp$
    Other tasks:
        This adversary makes other arbitary tasks.
```

Figure 7.28: Code for Adversary for two Annonymous Channel, Adv_{2AC} (Part II)

Code for ideal Initiator $\overline{\text{Init}_i}(i \in \{1,2\})$ of Two Anonymous Channel, $\overline{\text{Init}_{2AC_i}}$

Signature:

$\texttt{sid}_{\texttt{2AC}} = (\{\overline{\text{Init}_1}, \overline{\text{Init}_2}\}, \overline{\text{Rec}}, \texttt{sid}'_{\texttt{2AC}})$

Input:	Output:
$in(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\overline{\text{Init}_i}}$	$send(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\text{F}_{2AC}}$
$in(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\overline{\text{Init}_i}}$	$send(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{F}_{2AC}}$
$in(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\overline{\text{Init}_i}}$	$send(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\text{F}_{2AC}}$

State:

$mes \in \{\bot, \top\}$, initially $\bot$ $\quad ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

$active \in \{\bot, \top\}$, initially $\bot$

Tasks:

$\{send(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\text{F}_{2AC}}, send(\texttt{Send}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{F}_{2AC}},$
$send(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\text{F}_{2AC}}\}*$

Figure 7.29: Code for ideal Initiator of Two Anonymous Channel, $\overline{\text{Init}_{2AC}}$ (Part I)

Code for ideal Initiator $\overline{\text{Init}_i}(i \in \{1,2\})$ of Two Anonymous Channel, $\overline{\text{Init}_{2AC_i}}$

```
Transitions:

    Establish Session:

        ESS1. $in(\mathtt{Establish_{2AC}}, \mathtt{sid_{2AC}})_{\overline{\text{Init}_i}}$
                pre: $active$ and $ntask = \perp$
                eff: $ntask :=$ ESS2

        ESS2. $send(\mathtt{Establish_{2AC}}, \mathtt{sid_{2AC}})_{\text{F}_{2AC}}$
                pre: $ntask =$ ESS2
                eff: $active := \top$ and $ntask := \perp$

    Data Sending Session:

        DSS1. $in(\mathtt{Send}, \mathtt{sid_{2AC}}, m)_{\overline{\text{Init}_i}}$
                pre: $active = \top$, $mes = \perp$ and $ntask = \perp$
                eff: $mes := m$ and $ntask :=$ DSS2

        DSS2. $send(\mathtt{Send}, \mathtt{sid_{2AC}}, m)_{\text{F}_{2AC}}$
                pre: $m := mes$ and $ntask =$ DSS2
                eff: $mes := \perp$ and $ntask := \perp$

    Expire Session:

        EXS1. $in(\mathtt{Expire_{2AC}}, \mathtt{sid_{2AC}})_{\overline{\text{Init}_i}}$
                pre: $active = \top$ and $ntask = \perp$
                eff: $ntask :=$ EXS2

        EXS2. $send(\mathtt{Expire_{2AC}}, \mathtt{sid_{2AC}})_{\text{F}_{2AC}}$
                pre: $ntask =$ EXS2
                eff: $active, mes$ and $ntask := \perp$
```

Figure 7.30: Code for ideal Initiator of Two Anonymous Channel, $\overline{\text{Init}_{2AC}}$ (Part II)

Code for ideal Receiver $\overline{\text{Rec}}$ of Two Anonymous Channel, $\overline{\text{Rec}_{2AC}}$

Signature:

$\texttt{sid}_{2AC} = (\{\overline{\text{Init}_1}, \overline{\text{Init}_2}\}, \overline{\text{Rec}}, \texttt{sid}'_{2AC})$

Input:	Output:
$\mathit{in}(\texttt{Establish}_{2AC}, \texttt{sid}_{2AC})_{\overline{\text{Rec}}}$	
$\mathit{receive}(\texttt{Response}, \texttt{sid}_{2AC}, \mathit{mes})_{F_{2AC}}$	$\mathit{out}(\texttt{Receive}, \texttt{sid}_{2AC}, m)_{\overline{\text{Rec}}}$
$\mathit{in}(\texttt{Expire}_{2AC}, \texttt{sid}_{2AC})_{\overline{\text{Rec}}}$	

State:

$\mathit{mes} \in (\{0,1\}^*) \cup \{\bot\}$ $\qquad$ $\mathit{ntask} \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

$\mathit{active} \in \{\bot, \top\}$, initially $\bot$

Tasks:

$\{\mathit{out}(\texttt{Receive}, \texttt{sid}_{2AC}, m)_{\text{Rec}}\}$

Figure 7.31: Code for ideal Receiver of Two Anonymous Channel, $\overline{\text{Rec}_{2AC}}$ (Part I)

Code for ideal Receiver $\overline{\text{Rec}}$ of Two Anonymous Channel, $\overline{\text{Rec}_{2AC}}$

```
Transitions:

    Establish Session:

        ESS1. $in(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\overline{\text{Rec}}}$
              pre: $active$ and $ntask = \perp$
              eff: $active :=$

    Data Sending Session:

        DSS1. $receive(\texttt{Response}, \texttt{sid}_{\texttt{2AC}}, m)_{\text{F}_{2AC}}$
              pre: $active = \top$, $mes$ and $ntask = \perp$
              eff: $mes := m$ and $ntask :=$ DSS2
        DSS2. $out(\texttt{Receive}, \texttt{sid}_{\texttt{2AC}}, m)_{\overline{\text{Rec}}}$
              pre: $ntask =$ DSS2
              eff: $mes$ and $ntask := \perp$

    Expire Session:

        EXS1. $in(\texttt{Expire}_{\texttt{2AC}}, \texttt{sid}_{\texttt{2AC}})_{\overline{\text{Rec}}}$
              pre: $active = \top$, $mes$ and $ntask = \perp$
              eff: $ntask :=$ EXS2
```

Figure 7.32: Code for ideal Receiver of Two Anonymous Channel, $\overline{\text{Rec}_{2AC}}$ (Part II)

Code for Simulator for Anonymous Channel, Sim_{2AC}

Signature:

$\mathtt{sid}_{\mathtt{2AC}} = (\{\overline{\mathrm{Init}_1}, \overline{\mathrm{Init}_2}\}, \overline{\mathrm{Rec}}, \mathtt{sid}'_{\mathtt{2AC}})$

Input:

$\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F}_{2AC}}$

$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{mes})_{\mathrm{F}_{2AC}}$

$\mathit{receive}(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2AC}}$

Output:

$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{F}_{2AC}}$

Other:

*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message and $\mathit{out}(*)$.

State:

$\mathit{active} \in \{\bot, \top\}$, initialy $\bot$ $\qquad$ $\mathit{smes} \in \{0,1\}^* \cup \{\bot\}$, initialy $\bot$

$\mathit{ntask} \in (\{0,1\}^*) \cup \{\bot\}$, initialy $\bot$

Other arbitrary variables; cal ”new” variables.

Tasks:

$\{\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, \mathit{ok})_{\mathrm{F}_{2AC}}\}$

Figure 7.33: Code fot Simulator for Anonymous Channel, Sim_{2AC} (Part I)

Code for Simulator for Anonymous Channel, Sim_{2AC}

Transitions:

Establish Session:

```
**ESS1.** $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{F}_{2AC}}$
        pre: $active = \bot$, $ntask = \bot$
        eff: $active := \top$
```

Data Sending Session:

```
**DSS1.** $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, mes)_{\mathrm{F}_{2AC}}$
        pre: $active = \top$ and $ntask = \bot$
        eff: $smes := mes$ and $ntask :=$ DSS2
**DSS2.** $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{2AC}}, ok)_{\mathrm{F}_{2AC}}$
        pre: $ntask =$ DSS2
        eff: $ntask := \bot$
```

Expire Session:

```
EXS1. $receive(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{X}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{X}}_{2AC}}$
        pre: $active = \top$   eff: $active := \bot$
```

Other tasks:

This simulator makes arbitrary tasks to simulate the real world protocol system Real_{2AC}. The tasks mey be run with the information obtained from the simulator. Additionaly, this simulator can output the message from the adversary of the simiulating world to the environment.

Figure 7.34: Code fot Simulator for Anonymous Channel, Sim_{2AC} (Part II)

7.3 Equivalence Between DIC and SC

In this section, we prove that the DIC is equivalent to the SC under a specific type of schedule. To prove this, we show two reductions, SC to DIC and DIC to SC. Here, we consider a one bit message exchange, i.e., $|m| = 1$. Informally, the reduction of SC to DIC is proven as described hereafter. To make the channel between Init and Rec secure, the parties exchange a random bit (as a secret shared key) using the DIC. The message encrypted using the shared key is exchanged using a public channel. Communications are conducted not using the DIC but by a public channel. When the next message is sent, the parties restart from the key exchange stage. Here, the key exchange takes place under the master schedule. After the key exchange, the cipher text generated by the secret key is sent. The other reduction of DIC to SC is proven as described hereafter. Parties Init and Rec exchange two messages using the SC. One message is m, the message that the sender wants to send. The other message is a dummy message to conceal the message direction. More specifically, sender Init sends message m and the receiver sends dummy message s under a specific type of schedule M. We generate random message s using F_{SRC}. Note that, the adversary cannot identify the direction of the message because the messages are exchanged under a specific type of schedule. In this section, we must consider the schedules (key exchange schedule and message exchange schedule) to avoid exposing information to an adversary. In the UC framework, all schedules are under control of an adversary. So, we use the task PIOA framework.

7.3.1 Reduction of DIC to SC

Let π'_{DIC} be a protocol of DIC. Let $M_{\pi'_{\text{DIC}}}$ be master schedule $M_{psync}(\text{Init}'_{\text{DIC}}.send(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}, \text{Rec}'_{\text{DIC}}.send(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}})$ for π'_{DIC}.

Let $\text{Init}'_{\text{DIC}}$ and Rec'_{DIC} be the initiator code and receiver code for a real system, see Fig.7.35, Fig.7.36 and Fig.7.37, and Fig.7.38 and Fig.7.40, respectively. Let $\overline{\text{Init}'_{\text{DIC}}}$ and $\overline{\text{Rec}'_{\text{DIC}}}$ be the initiator code identical to $\overline{\text{Init}_{\text{DIC}}}$

and receiver code identical to $\overline{\text{Rec}_{\text{DIC}}}$ for an ideal system, respectively. Finally, let Adv'_{DIC} and Sim'_{DIC} be the adversary code and the simulator code in Fig.7.41 and Fig.7.42, and Fig.7.43 and Fig.7.44, respectively. Let $\text{Real}'_{\text{DIC}}$ and $\text{Ideal}'_{\text{DIC}}$ be a DIC protocol system and a DIC functionality system defined, respectively, as follows:

$$\text{Real}'_{\text{DIC}} := \mathit{hide}(\text{Init}'_{\text{DIC}}||\text{Rec}'_{\text{DIC}}||\text{Adv}'_{\text{DIC}}||\text{F}_{\text{SRC}}||\text{F}_{\text{SC}}, \{\mathit{rand}(*)\}),$$
$$\text{Ideal}'_{\text{DIC}} := \overline{\text{Init}'_{\text{DIC}}}||\overline{\text{Rec}'_{\text{DIC}}}||\text{Sim}'_{\text{DIC}}||\text{F}_{\text{DIC}}.$$

Tasks $\overline{\text{Init}'_{\text{DIC}}}$ and $\overline{\text{Rec}'_{\text{DIC}}}$ relay the input messages from the environment to the ideal functionality task and relay the received messages from the ideal functionality task to the environment, respectively, as interface parties in the ideal system.

Theorem 6. *DIC protocol system* $\text{Real}'_{\text{DIC}}$ *perfectly hybrid-implements DIC functionality system* $\text{Ideal}'_{\text{DIC}}$ *with respect to an adaptive adversary under master schedule* $M_{psync}(\mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}, \mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}})$*. (DIC is reducible to SC with respect to an adaptive adversary under master schedule* $M_{psync}(\mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}, \mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}})$*).*

Let ϵ_{R} and ϵ_{I} be discrete probability measures on finite executions of $\text{Real}'_{\text{DIC}}||\text{Env}$ and $\text{Ideal}'_{\text{DIC}}||\text{Env}$, respectively. We prove Theorem 6 by showing that ϵ_{R} and ϵ_{I} satisfy the trace distribution property, $\mathit{tdist}(\epsilon_{\text{R}}) = \mathit{tdist}(\epsilon_{\text{I}})$. Here, we define correspondence R between the states in $\text{Real}'_{\text{DIC}}||\text{Env}$ and the states in $\text{Ideal}'_{\text{DIC}}||\text{Env}$. We say $(\epsilon_{\text{R}}, \epsilon_{\text{I}}) \in R$ if and only if for every $s \in \text{supp.lst}(\epsilon_{\text{R}})$ and $u \in \text{supp.lst}(\epsilon_{\text{I}})$, all of the state correspondences in Tables 7.17, 7.18, and 7.19 hold. We then prove R is a simulation relation in Lemma 3.

Lemma 3. *Relation R defined above is a simulation relation from* $\text{Real}'_{\text{DIC}}||\text{Env}$ *to* $\text{Ideal}'_{\text{DIC}}||\text{Env}$ *under master schedule* $M_{\pi'_{\text{DIC}}}$.

Proof. We prove that R is a simulation relation from $\text{Real}'_{\text{DIC}}||\text{Env}$ to $\text{Ideal}'_{\text{DIC}}||\text{Env}$ using mapping corrtask $R^*_{\text{Real}'_{\text{DIC}}||\text{Env}} \times R_{\text{Real}'_{\text{DIC}}||\text{Env}} \rightarrow R^*_{\text{Ideal}'_{\text{DIC}}||\text{Env}}$, which is defined hereafter.

For any $(\rho, T) \in (R^*_{\text{Real}'_{\text{DIC}}||\text{Env}} \times R_{\text{Real}'_{\text{DIC}}||\text{Env}})$, the state correspondences in 7.17 and 7.18 hold.

The task sequence of system $\text{Real}'_{\text{DIC}}||\text{Env}$ are perfectly corresponds to that for system $\text{Ideal}'_{\text{DIC}}||\text{Env}$ under the schedule $M_{\pi'_{\text{DIC}}}$. Formally, to prove that R is a simulation relation from $\text{Real}'_{\text{DIC}}||\text{Env}$ to $\text{Ideal}'_{\text{DIC}}||\text{Env}$, we show that R satisfies the start condition and step condition.

- **Start condition**
  It is true that the start states of s and u in $\text{Real}'_{\text{DIC}}||\text{Env}$ and $\text{Ideal}'_{\text{DIC}}||\text{Env}$, respectively, are on the Dirac measures. That is, the start states of s and u satisfy relation R because the start states of s and u are all $\perp$ for each task on master schedule $M_{\pi'_{\text{DIC}}}$. Therefore, the trace distribution property holds.

- **Step condition**
  Let $\epsilon'_{\text{R}} = apply(\epsilon_{\text{R}}, T)$ and $\epsilon'_{\text{I}} = apply(\epsilon_{\text{I}}, corrtasks(\rho, T))$. If $(\epsilon_{\text{R}}, \epsilon_{\text{I}}) \in R$, $\rho \in R^*_{\text{Real}'_{\text{DIC}}||\text{Env}}$ and ϵ_{R} is consistent with ρ, then ϵ_{I} is consistent with $full(corrtasks)(\rho)$ and $T \in \text{Real}'_{\text{DIC}}||\text{Env}$. Then there exist the following.

  - Probability measure p on countable index set I,
  - Probability measures $\epsilon'_{\text{R},j}$, $j \in I$, on finite executions of $\text{Real}'_{\text{DIC}}||\text{Env}$, and
  - Probability measures $\epsilon'_{\text{I},j}$, $j \in I$, on finite executions of $\text{Ideal}'_{\text{DIC}}||\text{Env}$,

  such that:

  - For each $j \in I$, $\epsilon'_{\text{R},j} \; R \; \epsilon'_{\text{I},j}$,
  - $\Sigma_{j \in I} p(j)(\epsilon'_{\text{R},j}) = apply(\epsilon_{\text{R}}, T)$, and
  - $\Sigma_{j \in I} p(j)(\epsilon'_{\text{I},j}) = apply(\epsilon_{\text{I}}, corrtask(\rho, T))$.

Task Correspondence

For any $(\rho, T) \in (R^*_{\mathrm{Real}'_{\mathrm{DIC}}\|\mathrm{Env}} \times R_{\mathrm{Real}'_{\mathrm{DIC}}\|\mathrm{Env}})$, the following task correspondence that is also summarized in Table 7.20 holds.

1. **Establish Session**

   (a) $\mathrm{Init}'_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
   $=_{\mathrm{corr.}}\ \overline{\mathrm{Init}'_{\mathrm{DIC}}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
   Let $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ be $\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$ and $\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$, respectively. We assume that for each state, $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. The precondition of $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ is $\mathit{ntask} = \mathrm{ESS2}$ from each code. $\mathtt{T}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is enabled (or disabled) in s (resp., u) if and only if $s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{ntask} = \mathrm{ESS2}$ (resp. $u.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.\mathit{ntask} = \mathrm{ESS2}$). From ($j$) in Table 7.17, the state correspondence implies that $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

      i. Disable Case:
      Let I and p be the set that has a single element and Dirac measure on I, respectively. Let $\epsilon'_{\mathrm{R},1} = \epsilon'_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon'_{\mathrm{I}}$. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, we obtain $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ from relation $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. The trace distribution equivalence property, $\mathit{tdist}(\epsilon'_{\mathrm{R}}) = \mathit{tdist}(\epsilon'_{\mathrm{I}})$, also holds since $\mathit{tdist}(\epsilon_{\mathrm{R}}) = \mathit{tdist}(\epsilon_{\mathrm{I}})$ under $M_{\pi'_{\mathrm{DIC}}}$.

      ii. Enable Case:
      Let q denote the state of preconditions $\mathit{ntask} = \mathrm{ESS2}$. Let $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ be the action enabled in q for each world. We show that each of $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ is a unique action that is enabled in q. From the definition of $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$, the precondition is only $\mathit{ntask} = \mathrm{ESS2}$, and is unique in all tasks in $\mathrm{Init}'_{\mathrm{DIC}}$ and $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$, respectively. Then, there are two unique effects that update the *active*

and *ntask* to be $\top$ and $\bot$, respectively. From the precondition and the effect of $\mathtt{T}_{\mathtt{REAL}}$, and the state equivalence of (`i`) and (`j`), we obtain that the subsequent of $\mathtt{T}_{\mathtt{REAL}}$ (and $\mathtt{T}_{\mathtt{IDEAL}}$) is also a unique action that is enabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

Let I and p be the set that has a single element and the Dirac measure on I, respectively. Let $\epsilon'_{\mathrm{R},1} = \epsilon'_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon'_{\mathrm{I}}$. Here, we establish the property of R for ϵ'_{R} and ϵ'_{I} to show that $(\epsilon'_{\mathrm{R}}, \epsilon'_{\mathrm{I}}) \in R$. Then we show trace distribution equivalence for ϵ'_{R} and ϵ'_{I}. To establish this property, consider any state $s' \in \mathrm{supp.lst}(\epsilon'_{\mathrm{R}})$ and $u' \in \mathrm{supp.lst}(\epsilon'_{\mathrm{I}})$. Let s be any state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ such that $s' \in \mathrm{supp}(\mu_{\mathrm{s}})$, where $(s, \zeta, \mu_s) \in \mathrm{Real}'_{\mathrm{DIC}} \| \mathrm{Env}$. Let u be any state in $\mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ such that $u' \in \mathrm{supp}(\mu_{\mathrm{u}})$, where $(u, corrtask(\rho, \zeta), \mu_u) \in \mathrm{Ideal}'_{\mathrm{DIC}} \| \mathrm{Env}$.

It is true that $\mathtt{T}_{\mathtt{REAL}}$ updates $\mathrm{Init}.active$ to $\top$ and $\mathrm{Init}'_{\mathrm{DIC}}.ntask$ to $\bot$ from the definition of the effect of $\mathtt{T}_{\mathtt{REAL}}$. Similarly, $\mathtt{T}_{\mathtt{IDEAL}}$ updates $\overline{\mathrm{Init}'_{\mathrm{DIC}}}.active$ to $\top$ and $\overline{\mathrm{Init}'_{\mathrm{DIC}}}.ntask$ to $\bot$ from the definition of the effect of $\mathtt{T}_{\mathtt{IDEAL}}$. From the states equivalences of (i) and (j) in Table 7.17, we have $u.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.active = s.\mathrm{Init}'_{\mathrm{DIC}}.active$ and $u.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.ntask = s.\mathrm{Init}'_{\mathrm{DIC}}.ntask$. We obtain that $u'.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.active = s'.\mathrm{Init}'_{\mathrm{DIC}}.active$ and $u'.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.ntask = s'.\mathrm{Init}'_{\mathrm{DIC}}.ntask$. By the definition of $\mathrm{Init}'_{\mathrm{DIC}}$ and $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$, $\mathtt{T}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is a unique action that updates the state of *active* of $\mathrm{Real}'_{\mathrm{DIC}}$ (resp., $\mathrm{Ideal}'_{\mathrm{DIC}}$). Therefore, we obtain that the trace distribution property $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(b) $\mathrm{Rec}'_{\mathrm{DIC}}.send(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}'_{\mathrm{DIC}}}.send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
This case is analogous to the case of 1a. The state correspondences are (m) and (n). This case is also uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

(c) $\mathrm{F}_{\mathrm{SC}}.send(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.send(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$

The precondition and effect of these tasks are identical to each other. The preconditions of the task on the left side of the equation are $active = \top$ and $ntask = \mathrm{ESS2}$. This is equivalent to the precondition for the task on the right side. The effect of the task on the left is $ntask := \bot$. This effect is also the same as that for the task on the right. Let $\mathtt{T_{REAL}}$ be $\mathrm{F_{SC}}.send(\mathtt{SID}, \mathtt{sid_{SC}})_{\mathrm{Adv}}$. Let $\mathtt{T_{IDEAL}}$ be $\mathrm{F_{DIC}}.send(\mathtt{SID}, \mathtt{sid_{DIC}})_{\mathrm{Adv}}$. We show that $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. We consider that for each state in $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. Then, $\mathtt{T_{REAL}}$ enables (or disables) in s if and only if $s.\mathtt{T_{REAL}}.active = \top$ and $s.\mathtt{T_{REAL}}.ntask = \mathrm{ESS2}$. The precondition of $\mathtt{T_{IDEAL}}$, (f) in Table 7.17, implies that $\mathtt{T_{IDEAL}}$ is uniformly enabled or disabled. The rest of this proof is similar to that for the task of 1a.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (f). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondence of pre: (f), and state correspondences of eff: (d) and (f), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

2. **Data Sending Session**

Here, we consider two cases for this session. One is that Env inputs $in(\mathtt{Send}, \mathtt{sid_{DIC}}, m)_{\mathrm{Init'}}$ to $\mathrm{Init}'_{\mathrm{DIC}}$ and message receiver $\mathrm{Rec}'_{\mathrm{DIC}}$ outputs message $out(\mathtt{Receive}, \mathtt{sid_{DIC}}, plain)_{\mathrm{Rec'}}$. The other is that Env inputs $in(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{Rec}'}$ to $\mathrm{Rec}'_{\mathrm{DIC}}$ and message receiver $\mathrm{Init}'_{\mathrm{DIC}}$ outputs message $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, plain)_{\mathrm{Init}'}$. The two cases are given the same consideration, so we hereafter consider the first case. The basic task sequences are in Table 7.21. Note that the simulation is also shown as the same sequences for Real Execution in Table 7.21. The task sequences for the Real Execution are correspond to those for the Ideal Execution.

The flow of the states in each task is shown in Tables 7.22, 7.23, and 7.24 for each world. From the initial and final values in Table 7.22, 7.23, and 7.24, we obtain the result of state equivalence in 7.17. That is, if the state equivalence in 7.17 holds before $\mathrm{Real}'_{\mathrm{DIC}}$ is enabled (or disabled), the state equivalence in 7.17 after $\mathrm{Real}'_{\mathrm{DIC}}$ is finished also holds.

(a) Disable Case: This is a trivial case because all the states of the parties are $\perp$. The states do not change before or after the protocol starts in each world. That is, Env inputs no message to any party. Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from the task task definition and the state correspondences. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(b) Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definition and the state flows in Tables 7.22, 7.23, and 7.24, it is clear that the initial state is the same as the final state for each task in each world. In addition, the states of the real task are also the same as those for the ideal world after the data sending session is executed. That is, $(a) \sim (l)$ in Table 7.25 hold. Therefore, we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

3. **Expire Session**

(a) $\mathrm{Init}'_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Init}'_{\mathrm{DIC}}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
The states of precondition and effect for $send(\mathtt{Expire}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F}_{\mathrm{SC}}}$ are the same as those for $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$) where $ntask$ = EXS2. That is, if (j) in Table 7.17 holds, then these tasks are enabled (or disabled) in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

   i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (j). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

   ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence of pre: (j), and state correspondences of eff: (i) and (j), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(b) $\mathrm{Rec}'_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}'_{\mathrm{DIC}}}, send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
The precondition and effect for the real task are the same as those for the ideal task. The precondition is only $ntask$ = EXS2 and the effects are $active := \perp$ and $ntask := \perp\dot{}$From (n) in Table 7.17, these tasks are enabled (or disabled) in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

   i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (n). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence of pre: (n), and state correspondences of eff: (m) and (n), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(c) $\mathrm{F}_{\mathrm{SC}}.send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$
$=_{\mathrm{corr.}}$ $\mathrm{F}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$
The precondition and effect for the real task are the same as those for the ideal task. The precondition is only $ntask = \mathrm{EXS2}$ and the effects are $active := \perp$ and $estcond_X := \perp$ for all X (and $estcond_{\mathrm{Init}} and estcond_{\mathrm{Rec}} := \perp$ in F_{SC}) and $ntask := \perp$. From (f) in Table 7.17, these tasks are enabled (or disabled) in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (f). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence of pre: (f), and state correspondences of eff: (a),(b),(d) and (f), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Environment Env

From the task definitions and state correspondence (o) in Table7.17, the provability measures of both tasks are uniformly enabled or disabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

> **Claim 1** The state of Env remains static in all states in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}). Let q_e denote this state of Env. This follows from state correspondence o.
>
> **Claim 2** If T is a task of Env, then T is either enabled or disabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}) (simultaneously). Furthermore, if T is enabled in all states in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}), then:
>
> 1. There exists unique action $a \in$ T that is enabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).
> 2. There exists a unique transition of Env from q_e with action a. Let $tr_e = (q_e, a, \mu_e)$ be this transition.

By considering Claim 7.3.1, task T of Env is uniformly enabled or disabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}). If T is disabled, let $I = 1$, we obtain $\epsilon'_{\mathrm{R},1} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon_{\mathrm{I}}$, and the result is $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ since we have $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. If T is enabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}), Claim 7.3.1 implies that there exists unique action a in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}) and transition tr_e of Env from q_e enabled with action a, where $tr_e = (q_e, a, \mu_e)$.

Non Corrupted Case:

1. a is an input / output action of Init. We assume that a is an input action such as $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}'}$ $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}'}$, $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}'}$, and $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, plain)_{\mathrm{Init}'}$.

   Let s be any state such that $s' \in \mathrm{supp}(\mu_{\mathrm{s}})$, where $(s, a, \mu_s) \in D_{\mathrm{Real}'_{\mathrm{DIC}} \| \mathrm{Env}}$. Let u be any state such that $u' \in \mathrm{supp}(\mu_{\mathrm{u}})$, where

$(u, a, \mu_u) \in D_{\mathrm{Ideal}'_{\mathrm{DIC}} \| \mathrm{Env}}$. For each a, we check that the state correspondences for s' and u' hold if those for s and u hold. If each a is input from Env, then the precondition and effect for the real task are exactly the same as those for the ideal task. For example, if ,the input message is $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}'}$, then the precondition is $active, ntask = \perp$ and the effect is $ntask := \mathrm{ESS2}$. These states for the real task correspond to those for the ideal task. So, in the case of the enabled (or disabled), it is hold that tate correspondences (o), (i), and (j) hold for s' and u', if the correspondences for s and u hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$. This result also works well for $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}'}$ and $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}'}$.

2. a is an input / output action of Rec. We assume that a is an input action such as $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}'}$, $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}'}$, $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}'}$ and $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, plain)_{\mathrm{Rec}'}$. This is analogous to 1.

3. a is an input action of Adv. This means that $a = input(g)_{\mathrm{Adv}}$ for some fixed g. For example, g is a corrupt message for some $party \in \{\mathrm{Init}, \mathrm{Rec}\}$. From the fact that the state correspondences $(A) \sim (U)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

4. a is an internal or an output action of Env. Task a in the real world is identical to that in the ideal world. From the fact that state correspondence (o) for s and u holds, we obtain that state correspondence (o) for s' and u' holds. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R},j}) = trace(\epsilon'_{\mathrm{I},j})$.

Corrupted Case:

1. a is an input action of Adv and $party \in \{\mathrm{Init}, \mathrm{Rec}\}$ Here, the party is included in the case of $\mathrm{Init} \wedge \mathrm{Rec}$. Let q_{Adv} be the state of Adv

or Sim, which is the same in all supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}). Let $tr_{\mathrm{Adv}} = (q_{\mathrm{Adv}}, a, \mu_{\mathrm{Adv}})$ be a transition of Adv with action a from q_{Adv}. From Claim 7.3.1, tr_{Adv} is a unique transition. Here, we suppose that supp(($\mu_{\mathrm{e}} \times \mu_{\mathrm{Adv}}$)) is the pair set $\{(q_{1,j}, q_{2,j}) : j \in I\}$, where I is a countable set. Let p be the probability measures such that for each j, $p(j) = (\mu_e \times \mu_{\mathrm{Adv}})(q_{1,j}, q_{2,j})$. For each j, let $\epsilon'_{\mathrm{R},j}$ be $\epsilon'_{1,j}(\alpha) = \epsilon_1(\alpha')$, where $\alpha \in$ supp(ϵ'_1) such that $\mathtt{lst}(\alpha)$.Env $= q_{1,j}$ and $\mathtt{lst}(\alpha)$.Adv $= q_{2,j}$. The $\epsilon'_{2,j}$ is analogously constructed from ϵ'_2.

The rest of this proof is the same as that for 1 by the state correspondences for each case $party \in \{\mathrm{Init}, \mathrm{Rec}, \mathrm{Init} \wedge \mathrm{Rec}\}$. Finally, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R},j}) = trace(\epsilon'_{\mathrm{I},j})$.

Adversary Adv

From the task definitions and state correspondences $(A) \sim (U)$ in Table 7.18, the provability measures for both tasks are uniformly enabled or disabled in every state in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}).

Claim 3 The state for Adv or Sim is the same in all states in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}). Let q_{Adv} denote this state of Adv and Sim. This follows from state correspondence of Sim.

Claim 4 If T is a task of Adv, then T is either enabled or disabled in every state in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}). Furthermore, if T is enabled in all states in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}), then:

1. There is unique action $a \in$ T that is enabled in every state in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}).
2. There is a unique transition of Adv from q_{Adv} with action a, and let $tr_{\mathrm{Adv}} = (q_{\mathrm{Adv}}, a, \mu_{\mathrm{Adv}})$ be this transition.

By considering Claim.7.3.1, task T of Adv is uniformly enabled or disabled in every state in supp.lst(ϵ_{R}) ∪ supp.lst(ϵ_{I}). If T is disabled, let $I = 1$, we obtain $\epsilon'_{\mathrm{R},1} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon_{\mathrm{I}}$, and this results in that $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$

since we have $\epsilon_R R \epsilon_I$. If T is enabled, T is enabled in every state in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I). Claim 7.3.1 implies that there is unique action a in every state in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I) and transition tr of Adv from q_e enabled with action a, where $tr_{\text{Adv}} = (q_{\text{Adv}}, a, \mu_{\text{Adv}})$. The following cases for the "Non Corrupted Case" and "Corrupted Case" can be considered.

Non Corrupted Case:

1. a is an input action of Env. From the fact that state correspondences $(A) \sim (U)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_R) = trace(\epsilon'_I)$.

2. a is an input or output action of functionality task. This case concerns message $receive(\mathtt{SID}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$, $receive(\mathtt{Send}, \mathtt{sid}_{\text{SC}}, |m|)_{\text{F}_{\text{SC}}}$, $receive(\mathtt{Expire}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$ and $send(\mathtt{Response}, \mathtt{sid}_{\text{SC}}, ok)_{\text{F}_{\text{SC}}}$. The rest of this proof is analogous to case 1. From the fact that state correspondences $(A) \sim (U)$ for s and u hold, we obtain that state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_R) = trace(\epsilon'_I)$.

3. a is either an output action of Adv that is not an input action of Env, Init, Rec or functionality task, or is an internal action of Adv. This case concerns "new" tasks. The rest of this proof is analogous to case 1. From the fact that the state correspondences $(A) \sim (U)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_R) = trace(\epsilon'_I)$.

4. a is an output action of $out(*)_{adv}$. This case is also works well although this action may affect Env. However, the transition of Env $tr_e = (q_e, a, \mu_e)$ is unique from Claim 7.3.1. Claim 7.3.1 also says that the state of Env remains static in all states in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I). This follows from state correspondence o. Similarly, from the def-

inition and some claims, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Corrupted Case:

This is the case that the static and adaptive adversary Adv corrupts $party \in \{\mathrm{Init}, \mathrm{Rec}\}$.

1. a is input/output action $in(*)_{party}$ and $out(*)_{party}$ of corrupted party, $party \in \{\mathrm{Init}, \mathrm{Rec}\}$. This case is also works well from Claim 7.3.1 and state correspondence in Table 7.17 ~ 7.19.

Perfect Simulation

The simulation of $\mathrm{Sim}'_{\mathrm{DIC}}$ is perfectly executed for the establish session, data sending session and expire session with respect to no corruption, static corruption and adaptive corruption by an adversary.

1. **No Corruption**

   (a) **Establish Session** First, in the establish session, environment Env sends establish message $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Init}'}}$ and message $in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$ to initiator $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ and receiver $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$, respectively. They send establish session messages $send(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. The functionality sends $send(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ to $\mathrm{Sim}'_{\mathrm{DIC}}$. After $\mathrm{Sim}'_{\mathrm{DIC}}$ receives the message , $\mathrm{Sim}'_{\mathrm{DIC}}$ generates the parties Init and Rec in his simulation world to generate the real world situation in which Init and Rec exchange messages using F_{SC}. $\mathrm{Sim}'_{\mathrm{DIC}}$ then generates the establish session in the simulation world. That is, he inputs messages, $in(\mathtt{Establish}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Init}}$ and $in(\mathtt{Establish}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Rec}}$, to Init and Rec, respectively. Finally, the parties establish SC in the simulation world.

   Simulation Policy

i. After receiving $\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F_{DIC}}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes the following simulation.

   A. $\mathrm{Sim}'_{\mathrm{DIC}}$ prepares dummy parties, $\mathrm{Init}'_{\mathrm{DIC}}$, $\mathrm{Rec}'_{\mathrm{DIC}}$, and Adv and ideal functionality task $\mathrm{F_{SC}}$.

   B. $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs messages, $\mathit{in}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Init}'_{\mathrm{DIC}}}$ and $\mathit{in}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Rec}'_{\mathrm{DIC}}}$, to $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$, respectively.

   C. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes $\mathrm{Init}'_{\mathrm{DIC}}$ (resp., $\mathrm{Rec}'_{\mathrm{DIC}}$) send message $\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$ to $\mathrm{F_{SC}}$.

   D. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes $\mathrm{F_{SC}}$ send $\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Adv}}$ to Adv.

Task Correspondence of Simulation

i. $\mathrm{Init}'_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$
   $=_{\text{corr.}} \mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$

   pre: $\mathit{ntask} = \mathrm{ESS2}$; (L);

   eff: $\mathit{active} := \top$ and $\mathit{ntask} := \bot$; $(K), (L)$;

ii. $\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$
   $=_{\text{corr.}} \mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$

   pre: $\mathit{ntask} = \mathrm{ESS2}$; (Q);

   eff: $\mathit{active} = \top$ and $\mathit{ntask} := \bot$; $(P), (Q)$;

iii. $\mathrm{F_{SC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Adv}'_{\mathrm{DIC}}}$
   $=_{\text{corr.}} \mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{F_{SC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Adv}'_{\mathrm{DIC}}}$

   pre: $\mathit{active} = \top$ and $\mathit{ntask} = \mathrm{ESS2}$; $(F), (H)$;

   eff: $\mathit{ntask} := \bot$; (H);

The state *active* of Init and *rec* becomes $\top$, then state correspondences (P) and (K) hold. If the adversary obtains the message $\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$ in the simulation world, *active* becomes $\top$ from (S) in Table 7.19. The simulation in the establish session of the real world is perfectly executed by $\mathrm{Sim}'_{\mathrm{DIC}}$. Finally, the parties establish SC in the simulation world.

(b) **Data Sending Session** Next, in the data sending session, Env sends message $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Init}'}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}}}$) to $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ (or $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$). $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ sends $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. $\mathrm{F}_{\mathrm{DIC}}$ then sends $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$ to $\mathrm{Sim}'_{\mathrm{DIC}}$. After receiving the message, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $simulation(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, mes)$ to mimic the data sending session in the real world. That is, he inputs message $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Init}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Rec}}$) to Init (or Rec) in the simulation world. This is executed as follows. First, if the message sender is Init, then the receiver generates a random message with $rand(t)_{tval}$. Next, both parties, that is, Init and Rec, send messages to each other. Init sends $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$ to Rec and Rec sends $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, t)_{\mathrm{F}_{\mathrm{SC}}}$ to Init. This is executed under master schedule $M_{\pi'_{\mathrm{DIC}}}$. If the master schedule does not work, then the random message exchange is not occurred. If so, the adversary can identify the directions that the messages were sent.

The policy of $\mathrm{Sim}'_{\mathrm{DIC}}$ is described hereafter.

Simulation Policy

i. After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes the following simulation.

   A. $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $random(*)$ and selects dummy message t where $|t| = 1$.

   B. $\mathrm{Sim}'_{\mathrm{DIC}}$ generates random bit $o \in \{0, 1\}$ by executing $random(*)$.

   C. If $o = 0$, then $\mathrm{Sim}'_{\mathrm{DIC}}$ sends $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Init}'_{\mathrm{DIC}}}$ to $\mathrm{Init}'_{\mathrm{DIC}}$ and $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, t)_{\mathrm{Init}'_{\mathrm{DIC}}}$ to $\mathrm{Rec}'_{\mathrm{DIC}}$. Else, $\mathrm{Sim}'_{\mathrm{DIC}}$ performs the opposite actions.

   D. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes $\mathrm{Init}'_{\mathrm{DIC}}$ send $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$ and makes $\mathrm{Rec}'_{\mathrm{DIC}}$ send $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, t)_{\mathrm{F}_{\mathrm{SC}}}$ at the same moment according to $M_{\pi'_{\mathrm{DIC}}}$.

E. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes F_{SC} receive $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, m)_{\mathrm{Init}'_{\mathrm{DIC}}}$ and makes F_{SC} send $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, m)_{\mathrm{Adv}}$ to Adv.

F. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes F_{SC} receive $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, t)_{\mathrm{Rec}'_{\mathrm{DIC}}}$ and makes F_{SC} send $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, t)_{\mathrm{Adv}}$ to Adv.

G. If F_{SC} receives $\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{SC}}, ok)_{\mathrm{F}_{\mathrm{SC}}}$ from Adv twice, $\mathrm{Sim}'_{\mathrm{DIC}}$ continues the following.

H. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes F_{SC} receive $\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{SC}}, ok)_{\mathrm{Adv}}$ and makes F_{SC} send $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{SC}}, t)_{\mathrm{Init}'_{\mathrm{DIC}}}$ and $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{SC}}, mes)_{\mathrm{Rec}'_{\mathrm{DIC}}}$ to $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$, respectively.

I. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$ receive message $\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{SC}}, t)_{\mathrm{F}_{\mathrm{SC}}}$ and receive the message $\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$, respectively.

J. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes $\mathrm{Rec}'_{\mathrm{DIC}}$ output $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{SC}}, m)_{\mathrm{Rec}'_{\mathrm{DIC}}}$.

ii. $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$.

The details of this task sequence are shown in Table 7.21. This task sequence is also simulated by $\mathrm{Sim}'_{\mathrm{DIC}}$. So, state correspondences $(A) \sim (L)$ and $(N) \sim (T)$ hold. That is, simulator $\mathrm{Sim}'_{\mathrm{DIC}}$ executes the above-mentioned process to mimic the real world. The state correspondence in Tables 7.18 and 7.19 work well. The key point of this simulation is as follows. To mimic the real world, the simulator executes the parties that execute the tasks in the real world. Moreover, not to distinguish the output trace, the simulator simulates the real world in his simulation world using task codes. In the real world, $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$ use SC withoutan adversary identifying the direction in which the message was sent under master schedule $M_{\pi'_{\mathrm{DIC}}}$. In the simulation world, $\mathrm{Sim}'_{\mathrm{DIC}}$ obtains the same output which $\mathrm{Adv}'_{\mathrm{DIC}}$ outputs in the real world by his simulation. That is, the trace distributions in each world are indistinguishable by Env. In other words, since each task correspondence and the state correspondence work well, the following property works well, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

Task Correspondence of Simulation

i. $\text{Init}'_{\text{DIC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, m)_{\text{F}_{\text{SC}}}$
$=_{\text{corr.}}$ $\text{Sim}'_{\text{DIC}}.\text{Init}'_{\text{DIC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, m)_{\text{F}_{\text{SC}}}$

pre: $m := \mathit{smes}$ and $\mathit{ntask} = \text{DSS2}$; (L);

eff: $\mathit{smes} := \perp$ and $\mathit{ntask} := \text{DSS4}$; $(I), (L)$;

ii. $\text{Rec}'_{\text{DIC}}.\mathit{rand}(t)_{\mathit{tval}\text{F}_{\text{SRC}}} =_{\text{corr.}} \text{Sim}'_{\text{DIC}}.\text{Rec}'_{\text{DIC}}.\mathit{rand}(t)_{\mathit{tval}\text{F}_{\text{SRC}}}$

pre: $\mathit{ntask} = \perp$; (Q);

eff: $\mathit{dummy} := \top$, $\mathit{smes} := t$ and $\mathit{ntask} := \text{DSS2}$; $(O), (Q)$;

iii. $\text{Rec}'_{\text{DIC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, m)_{\text{F}_{\text{SC}}}$
$=_{\text{corr.}}$ $\text{Sim}'_{\text{DIC}}.\text{Rec}'_{\text{DIC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, m)_{\text{F}_{\text{SC}}}$

pre: $m := \mathit{smes}$ and $\mathit{ntask} = \text{DSS2}$; (Q);

eff: $\mathit{smes} := \perp$ and $\mathit{ntask} := \text{DSS4}$; $(N), (Q)$;

iv. $\text{F}_{\text{SC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, |m|)_{\text{Adv}}$
$=_{\text{corr.}}$ $\text{Sim}'_{\text{DIC}}.\text{F}_{\text{SC}}.\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, |m|)_{\text{Adv}}$

pre: $\mathit{active} = \top$, $\mathit{mes} \neq \perp$, $\mathit{okcond}_{\text{Adv}} = \perp$, $m := \mathit{mes}$ and $\mathit{ntask} = \text{DSS2}$; $(E) \sim (G), (H)$;

eff: $\mathit{ntask} := \text{DSS3}$; (H);

v. $\text{Adv}'_{\text{DIC}}.\mathit{send}(\texttt{Response}, \texttt{sid}_{\texttt{SC}}, \mathit{ok})_{\text{F}_{\text{SC}}}$
$=_{\text{corr.}}$ $\text{Sim}_{\text{SC}}.\text{Adv}'_{\text{DIC}}.\mathit{send}(\texttt{Response}, \texttt{sid}_{\texttt{SC}}, \mathit{ok})_{\text{F}_{\text{SC}}}$

pre: $\mathit{ntask} = \text{DSS2}$; (T);

eff: $\mathit{ntask} := \perp$; (T);

vi. $\text{F}_{\text{SC}}.\mathit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{SC}}, \mathit{mes})_{\text{Rec}'_{\text{DIC}}}$
$=_{\text{corr.}}$ $\text{Sim}_{\text{SC}}.\text{F}_{\text{SC}}.\mathit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, \mathit{mes})_{\text{Rec}'_{\text{DIC}}}$

pre: $\mathit{active}, \mathit{mes} = \top$, $\mathit{okcond}_{\text{Adv}} \neq \perp$ and $\mathit{ntask} = \text{DSS4}$; $(E), (F), (H)$;

eff: mes and $\mathit{okcond}_{\text{Adv}}, \mathit{ntask} := \perp$; $(E), (G), (H)$;

vii. $\text{F}_{\text{SC}}.\mathit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{SC}}, \mathit{mes})_{\text{Init}'_{\text{DIC}}}$
$=_{\text{corr.}}$ $\text{Sim}_{\text{SC}}.\text{F}_{\text{SC}}.\mathit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, \mathit{mes})_{\text{Init}'_{\text{DIC}}}$

pre: $\mathit{active}, \mathit{mes} = \top$, $\mathit{okcond}_{\text{Adv}} \neq \perp$ and $\mathit{ntask} = \text{DSS4}$; $(E), (F), (H)$;

eff: mes and $okcond_{\mathrm{Adv}}, ntask := \perp$; $(E), (G), (H)$;

viii. $\mathrm{Rec}'_{\mathrm{DIC}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Rec}'_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}'_{\mathrm{DIC}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Rec}'_{\mathrm{DIC}}}$

pre: $plain := rmes$ and $ntask = \mathrm{DSS5}$; $(O), (Q)$;

eff: $dummy, rmes$ and $ntask := \perp$; $(O), (R), (Q)$;

(c) **Expire Session** Finally, in the expire session, Env sends message $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Init}'}}$ and $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\overline{\mathrm{Rec}}}$ to $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$, respectively. They relay message $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. After receiving $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ from $\mathrm{F}_{\mathrm{DIC}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ terminates the session in the simulation world.

Simulation Policy

i. After receiving $receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes the following simulation.

A. $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs messages, $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}}$ and $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}}$, to $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$, respectively.

B. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes $\mathrm{Init}'_{\mathrm{DIC}}$ (resp., $\mathrm{Rec}'_{\mathrm{DIC}}$) send message $send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$ to F_{SC}.

C. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes F_{SC} send $send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$ to Adv.

D. $\mathrm{Sim}'_{\mathrm{DIC}}$ finishes the simulation of the expire session.

That is, $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs messages $in(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Init}}$ and $in(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Rec}}$ to Init and Rec in the simulation world. We assume that the state correspondences in Table 7.18 and 7.19 hold. From 3a, 3b and 3c, the state correspondences also hold after the simulation by $\mathrm{Sim}'_{\mathrm{DIC}}$. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Task Correspondence of Simulation

i. $\mathrm{Init}'_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Init}'_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$

pre: $ntask = \text{EXS2}$; (L);

eff: $active$ and $ntask := \perp$; $(K), (L)$;

ii. $\text{Rec}'_{\text{DIC}}.send(\texttt{Expire}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{F}_{\text{SC}}}$
$=_{\text{corr.}} \text{Sim}'_{\text{DIC}}.\text{Rec}'_{\text{DIC}}.send(\texttt{Expire}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{F}_{\text{SC}}}$

pre: $ntask = \text{EXS2}$; (Q);

eff: $active$ and $ntask := \perp$; $(P), (Q)$;

iii. $\text{F}_{\text{SC}}.send(\texttt{Expire}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{Adv}}$
$=_{\text{corr.}} \text{Sim}'_{\text{DIC}}.\text{F}_{\text{SC}}.send(\texttt{Expire}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{Adv}}$

pre: $ntask = \text{EXS2}$; (I);

eff: $active$, $estcond_X$ and $ntask := \perp$ for all X ; $(C) \sim (F)$, (H);

2. **Static Corruption**

This type of corruption is divided into the following three cases: only Init is corrupted by Adv, only Rec is corrupted by Adv, and both parties are corrupted by Adv. Once the corruption occurs, the adversary can identify the direction. However, the simulator can simulate all the cases, so Env can not distinguish the real world from the ideal world.

(a) Only Init is corrupted by Adv

This case means that Adv'_{DIC} corrupts only Init before the protocol starts. Adv'_{DIC} and Sim'_{DIC} identify the direction that the message was sent from Init to Rec and from Rec to Init, respectively. So, the simulation is perfectly executed.

i. After receiving the corrupt message from Env, Sim'_{DIC} prepares the situation in which only $\text{Init}'_{\text{DIC}}$ is corrupted and adds the following policy before 1(a)iB. Sim'_{DIC} makes Adv corrupt $\text{Init}'_{\text{DIC}}$.

ii. After receiving $receive(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{F}_{\text{DIC}}}$ in $party \in \{\text{Init}, \text{Rec}\}$, Sim'_{DIC} executes the following simulation.

A. If the message is input to corrupted party Init, $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init.

B. Else the message is input to Rec and $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec.

C. The remaining steps are the same as the simulation for the No Corrupted Case.

iii. After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Init}'_{\mathrm{DIC}}}}$.

If the protocol executes the establish session, data sending session, and expire session, in any case, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by $\mathrm{Sim}'_{\mathrm{DIC}}$. From the Task Correspondence in 7.3.1, the state correspondence 7.17, 7.18, and 7.19 hold in this case. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$ holds.

(b) Only Rec is corrupted by Adv

This case is analogous to case **??**. This case means that $\mathrm{Adv}'_{\mathrm{DIC}}$ corrupts only Rec before the protocol starts. $\mathrm{Adv}'_{\mathrm{DIC}}$ and $\mathrm{Sim}'_{\mathrm{DIC}}$ identify the direction that the message was sent from Init to Rec and from Rec to Init, respectively. So, the simulation is perfectly executed.

i. After receiving the corrupt message from Env, $\mathrm{Sim}'_{\mathrm{DIC}}$ prepares a situation in which only $\mathrm{Rec}'_{\mathrm{DIC}}$ is corrupted and adds the following policy before 1(a)iB. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes Adv corrupt $\mathrm{Rec}'_{\mathrm{DIC}}$.

ii. After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ from $\mathrm{F_{DIC}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes the following simulation.

A. If the message is input to corrupted party Rec, $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec.

B. Else the message is input to Init and $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init.

C. The remaining steps are the same as the simulation for the No Corrupted Case.

iii. After receiving $\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\overline{\mathrm{Rec}'_{\mathrm{DIC}}}}$.

If the protocol executes the establish session, data sending session and expire session, in any case, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by $\mathrm{Sim}'_{\mathrm{DIC}}$. From the Task Correspondence in 7.3.1, the state correspondences in 7.17, 7.18, and 7.19 hold in this case. That is, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$ holds.

(c) Both parties are corrupted by Adv

This case is also analogous to case **??**. This case means that $\mathrm{Adv}'_{\mathrm{DIC}}$ corrupts both Init and Rec before the protocol starts. $\mathrm{Adv}'_{\mathrm{DIC}}$ and $\mathrm{Sim}'_{\mathrm{DIC}}$ identify the direction that the message was sent from Init to Rec and from Rec to Init, respectively. So, the simulation is perfectly executed.

i. After receiving the corrupt message from Env, $\mathrm{Sim}'_{\mathrm{DIC}}$ prepares the situation in which only $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$ are corrupted and adds the following policy before 1(a)iB. $\mathrm{Sim}'_{\mathrm{DIC}}$ makes Adv corrupt $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$.

ii. If the data sending message is input to $\mathit{party} \in \{\mathrm{Init}, \mathrm{Rec}\}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs $\mathit{in}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathit{party}}$ to party.

iii. The remaining is the same as the simulation for the No Corrupted Case.

iv. After receiving $\mathit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in party, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathit{party}}$.

If the protocol executes the establish session, data sending session and expire session, in any case, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by $\mathrm{Sim}'_{\mathrm{DIC}}$. From the Task Correspondence in 7.3.1, the state correspondences 7.17, 7.18, and 7.19 hold in this case. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$ holds.

3. **Adaptive Corruption**

   In this case, the adversary corrupts some parties when he wants to do so at any time. We assume that the adversary corrupts the parties. This case is also simulated by the simulator, but the direction that the message was sent cannot be concealed from the adversary after he corrupts some parties. However, this case is also simulated by simulator $\mathrm{Sim}'_{\mathrm{DIC}}$, so the simulation is perfectly executed. This case is separated into the following cases.

   (a) **Establish Session**

   Instance 1: Before $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$ are activated.
   This case is analogous to case 2 because there is no secret information. The adversary can corrupt $\mathrm{Init}'_{\mathrm{DIC}}$, $\mathrm{Rec}'_{\mathrm{DIC}}$, or both, but the simulator can also corrupt the corresponding parties. This case is also perfectly simulated by $\mathrm{Sim}'_{\mathrm{DIC}}$.
   Instance 2: After $\mathrm{Init}'_{\mathrm{DIC}}$ is activated but before $\mathrm{Rec}'_{\mathrm{DIC}}$ is activated.
   This case is analogous to case 2 because there is no secret information. The adversary can corrupt $\mathrm{Init}'_{\mathrm{DIC}}$, $\mathrm{Rec}'_{\mathrm{DIC}}$, or both, but the simulator can also corrupt the corresponding parties. This case is perfectly simulated by $\mathrm{Sim}'_{\mathrm{DIC}}$.
   Instance 3: After $\mathrm{Rec}'_{\mathrm{DIC}}$ is activated but before $\mathrm{Init}'_{\mathrm{DIC}}$ is activated.
   This case is analogous to case 2 because there is no secret information. The adversary can corrupt $\mathrm{Init}'_{\mathrm{DIC}}$, $\mathrm{Rec}'_{\mathrm{DIC}}$, or both, but the simulator can also corrupt the corresponding parties. This case is also perfectly simulated by $\mathrm{Sim}'_{\mathrm{DIC}}$.
   Instance 4: After $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$ are activated.
   This case is analogous to case 2 because there is no secret information. The adversary can corrupt $\mathrm{Init}'_{\mathrm{DIC}}$, $\mathrm{Rec}'_{\mathrm{DIC}}$, or

both, but the simulator can also corrupt the corresponding parties. These case are also perfectly simulated by $\mathrm{Sim}'_{\mathrm{DIC}}$.

(b) **Data Sending Session**

Instance 1: Before or after activating $\mathrm{Init}'_{\mathrm{DIC}}$ or $\mathrm{Rec}'_{\mathrm{DIC}}$ by receiving $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}'_{\mathrm{DIC}}}$ or $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m$ $)_{\mathrm{Rec}'_{\mathrm{DIC}}}$, respectively, from the Env.
Env can execute only the message sending indication and the corrupt indication. So, in this case only the adversary corrupts the party. This case is also simulated by $\mathrm{Sim}'_{\mathrm{DIC}}$, because there is no secret information. So, the task corresponding works well and there exists a simulation relation between the real world and ideal world. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$ holds.

(c) **Expire Session**

Instance 1: After $\mathrm{Init}'_{\mathrm{DIC}}$ or $\mathrm{Rec}'_{\mathrm{DIC}}$ is activated with expire message.
Once the expire message is sent to $\mathrm{Init}'_{\mathrm{DIC}}$ or $\mathrm{Rec}'_{\mathrm{DIC}}$ by Env, this session terminates in the real world and ideal world. So the adversary can corrupt the parties. That is, this case is identical to case 2.

Simulation Policy

$\mathrm{Sim}'_{\mathrm{DIC}}$ simulates in his simulation world as follows:

(a) After receiving "corrupt $\mathrm{Init}'_{\mathrm{DIC}}$" message from Env,

- $\mathrm{Sim}'_{\mathrm{DIC}}$ corrupts $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ and checks whether $party \in \{\mathrm{Init}, \mathrm{Rec}\}$ has already sent the data sending message to the other party. If the message was already sent, $\mathrm{Sim}'_{\mathrm{DIC}}$ does the following. Else, $\mathrm{Sim}'_{\mathrm{DIC}}$ makes Adv corrupt Init.
- If $party = \mathrm{Init}$,

    - If $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Init, immediately.
    - Else, $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Rec, immediately.
  - Else, *party* = Rec,
    - If $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Rec, immediately.
    - Else, $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Init, immediately.
  - If more data sending messages are input to *party* from Env after $\mathrm{Sim}'_{\mathrm{DIC}}$ corrupts *party*, $\mathrm{Sim}'_{\mathrm{DIC}}$ can also simulate the situation. If the message is input to corrupted Init, $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs the sending message to corrupted *party* in his simulation. Else, the message is input to Rec and $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs the sending message to non-corrupted *party* in his simulation.
  - After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Init}'_{\mathrm{DIC}}}}$.

(b) After receiving the "corrupt $\mathrm{Rec}'_{\mathrm{DIC}}$" message from Env,

  - $\mathrm{Sim}'_{\mathrm{DIC}}$ corrupts $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$ and checks whether *party* $\in \{\mathrm{Init}, \mathrm{Rec}\}$ has already sent the data sending message to the other party. If the message was already sent, do as follows. Else, makes Adv corrupt Rec.
  - If *party* = Init,
    - If $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Rec, immediately.

    - Else, $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Init.

- Else, *party* = Rec,
    - If $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}}$ to Init in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Init.
    - Else, $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}}$ to Rec in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that Adv corrupts Rec.
- If more data sending messages are input in *party* from Env after $\mathrm{Sim}'_{\mathrm{DIC}}$ corrupts *party*, $\mathrm{Sim}'_{\mathrm{DIC}}$ can also simulate the situation. If the message is input to corrupted Init, $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs the sending message in non-corrupted *party* in his simulation. Else, the message is input to Rec and $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs the sending message to corrupted *party* in his simulation.
- After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$ in $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\overline{\mathrm{Rec}'_{\mathrm{DIC}}}}$.

(c) After receiving the "corrupt $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$" message from Env,

- $\mathrm{Sim}'_{\mathrm{DIC}}$ corrupts $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$ and checks whether *party* $\in \{\mathrm{Init}, \mathrm{Rec}\}$ has already sent the data sending message to the other party.

  If the message was already sent, $\mathrm{Sim}'_{\mathrm{DIC}}$ makes Adv corrupt Init and Rec and does the following. Else, $\mathrm{Sim}'_{\mathrm{DIC}}$ makes Adv corrupt Init and Rec.
    - If $\mathrm{Sim}'_{\mathrm{DIC}}$ has already input message sending request $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{party}$ to *party* in his simulation, then $\mathrm{Sim}'_{\mathrm{DIC}}$ simulates that $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs more data sending requests to the corresponding *party*. That is, if the party

that $\mathrm{Sim}'_{\mathrm{DIC}}$ has already sent a request message is equal to the party that received a message from Env, $\mathrm{Sim}'_{\mathrm{DIC}}$ inputs more data sending requested to the party.

  – Else, the input party in the simulation world is not same as the input party in the ideal world, $\mathrm{Sim}'_{\mathrm{DIC}}$ regards the input party in the simulation world as the input party which has already input a message in the ideal world. The other party in the simulation world is also regarded as the party which has not input a message yet in the ideal world.

- After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$ in *party*, $\mathrm{Sim}'_{\mathrm{DIC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathit{party}}$.

Whenever $\mathrm{Adv}'_{\mathrm{DIC}}$ corrupts some party, $\mathrm{Sim}'_{\mathrm{DIC}}$ corrupts the corresponding dummy party in the ideal world and forwards the obtained information to the simulated copy of $\mathrm{Adv}'_{\mathrm{DIC}}$. If $\mathrm{Adv}'_{\mathrm{DIC}}$ corrupts party $\mathrm{Init}'_{\mathrm{DIC}}$ or $\mathrm{Rec}'_{\mathrm{DIC}}$ then $\mathrm{Sim}'_{\mathrm{DIC}}$ corrupts $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ or $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$ in the ideal world, and provides a simulated copy of $\mathrm{Adv}'_{\mathrm{DIC}}$ in the simulation world with the states of the corrupted party. Conversely, $\mathrm{Sim}'_{\mathrm{DIC}}$ may obtain information from the simulated world with the corruption. Additionally, in this protocol party there is no secret information because F_{SC} is securely executed. In all cases, since $\mathrm{Sim}'_{\mathrm{DIC}}$ can simulate $\mathrm{Adv}'_{\mathrm{DIC}}$ by using his simulated world, Env cannot distinguish real world from ideal world. That is, simulating party corruption is perfectly executed.

Finally, relation R is a simulation relation from the task and state correspondence. We obtain Lemma3. □

Next, Theorem6 is obtained from Lemma3 immediately.

Proof. From Lemma 3 and Theorem 3, Theorem 6 is proved. That is, the trace distribution property, $\mathit{tdist}(\epsilon_{\mathrm{R}}) = \mathit{tdist}(\epsilon_{\mathrm{I}})$ holds with respect to adaptive adversary.

As a result, the simulation is perfectly executed because $\mathrm{Sim}'_{\mathrm{DIC}}$ can simulate the real world from the information message through $\mathrm{Adv}'_{\mathrm{DIC}}$. The tasks of the real world perfectly correspond with the tasks of the ideal world. That is,

$$\mathrm{Real}'_{\mathrm{DIC}} \| \mathrm{Env}\ \ \mathrm{Hyb.} \leq_0^{\mathtt{M}_{\pi'_{\mathrm{DIC}}}} \mathrm{Ideal}'_{\mathrm{DIC}} \| \mathrm{Env}.$$

□

Functionality

(a)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Init}} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Init}}$
(b)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Rec}} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Rec}}$
(c)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{okcond}_{\mathrm{Adv}'} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{okcond}_{\mathrm{Adv}'}$
(d)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{active}$
(e)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{mes} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{mes}$
(f)	$u.\mathrm{F}_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{ntask}$

Initiator

(g)	$u.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.\mathit{smes} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{smes}$
(h)	$u.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.\mathit{rmes} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{rmes}$
(i)	$u.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.\mathit{active} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{active}$
(j)	$u.\overline{\mathrm{Init}'_{\mathrm{DIC}}}.\mathit{ntask} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{ntask}$

Receiver

(k)	$u.\overline{\mathrm{Rec}'_{\mathrm{DIC}}}.\mathit{smes} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{smes}$
(l)	$u.\overline{\mathrm{Rec}'_{\mathrm{DIC}}}.\mathit{rmes} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{rmes}$
(m)	$u.\overline{\mathrm{Rec}'_{\mathrm{DIC}}}.\mathit{active} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{active}$
(n)	$u.\overline{\mathrm{Rec}'_{\mathrm{DIC}}}.\mathit{ntask} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{ntask}$

Environment

(o)	$u.\mathrm{Env} = s.\mathrm{Env}$

Table 7.17: State Correspondence for $\mathrm{Real}'_{\mathrm{DIC}}$ and $\mathrm{Ideal}'_{\mathrm{DIC}}$ (Part I)

Simulator (or Adversary)

(A)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{active}$
(B)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{ntask}$
(C)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Init}} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Init}}$
(D)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Rec}} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Rec}}$
(E)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{F}_{\mathrm{SC}}.\mathit{okcond}_{\mathrm{Adv}'} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{okcond}_{\mathrm{Adv}'}$
(F)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{F}_{\mathrm{SC}}.\mathit{active} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{active}$
(G)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{F}_{\mathrm{SC}}.\mathit{mes} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{mes}$
(H)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{F}_{\mathrm{SC}}.\mathit{ntask} = s.\mathrm{F}_{\mathrm{SC}}.\mathit{ntask}$
(I)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{smes} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{smes}$
(J)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{rmes} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{rmes}$
(K)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{active}$
(L)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{ntask}$
(M)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{dummy} = s.\mathrm{Init}'_{\mathrm{DIC}}.\mathit{dummy}$

Table 7.18: State Correspondence for $\mathrm{Real}'_{\mathrm{DIC}}$ and $\mathrm{Ideal}'_{\mathrm{DIC}}$ (Part II)

Simulator (or Adversary)

(N)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{smes} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{smes}$
(O)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{rmes} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{rmes}$
(P)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{active}$
(Q)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{ntask}$
(R)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{dummy} = s.\mathrm{Rec}'_{\mathrm{DIC}}.\mathit{dummy}$
(S)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{active} = s.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{active}$
(T)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{ntask} = s.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{ntask}$
(U)	$u.\mathrm{Sim}'_{\mathrm{DIC}}.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{length} = s.\mathrm{Adv}'_{\mathrm{DIC}}.\mathit{length}$

Table 7.19: State Correspondence for $\mathrm{Real}'_{\mathrm{DIC}}$ and $\mathrm{Ideal}'_{\mathrm{DIC}}$ (Part III)

1. **Establish Session**

(a)	$\text{Init}'_{\text{DIC}}.\textit{send}(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
	$=_{\text{corr.}} \overline{\text{Init}'_{\text{DIC}}}.\textit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
(b)	$\text{Rec}'_{\text{DIC}}.\textit{send}(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
	$=_{\text{corr.}} \overline{\text{Rec}'_{\text{DIC}}}, \textit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
(c)	$\text{F}_{\text{SC}}.\textit{send}(\texttt{SID}, \texttt{sid}_{\text{SC}})_{\text{Adv}} =_{\text{corr.}} \text{F}_{\text{DIC}}.\textit{send}(\texttt{SID}, \texttt{sid}_{\texttt{DIC}})_{\text{Adv}}$

2. **Expire Session**

(a)	$\text{Init}'_{\text{DIC}}.\textit{send}(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
	$=_{\text{corr.}} \overline{\text{Init}'_{\text{DIC}}}.\textit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
(b)	$\text{Rec}'_{\text{DIC}}.\textit{send}(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
	$=_{\text{corr.}} \overline{\text{Rec}'_{\text{DIC}}}, \textit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
(c)	$\text{F}_{\text{SC}}.\textit{send}(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{Adv}}$
	$=_{\text{corr.}} \text{F}_{\text{DIC}}.\textit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{Adv}}$

3. **Environment**

(a)	All tasks of environment Env in $\text{Real}'_{\text{DIC}}$ correspond to the tasks of environment in $\text{Ideal}'_{\text{DIC}}$.

Table 7.20: Corresponding Tasks for $\text{Real}'_{\text{DIC}}$ and $\text{Ideal}'_{\text{DIC}}$

Real Execution

No.	$\text{Init}'_{\text{DIC}}$	Rec'_{DIC}	F_{SC}	Adv'_{DIC}
1	$\mathit{in}(\texttt{Send},\texttt{sid}_{\texttt{DIC}},m)_{\text{Init}'_{\text{DIC}}}$			
2		$\mathit{rand}(t)_{tval}$ from $\text{F}_{\texttt{SRC}}$		
3	$\mathit{send}(\texttt{Send},\texttt{sid}_{\text{SC}},m)_{\text{F}_{\text{SC}}}$	$\mathit{send}(\texttt{Send},\texttt{sid}_{\text{SC}},t)_{\text{F}_{\text{SC}}}$		
4			$\mathit{receive}(\texttt{Send},\texttt{sid}_{\text{SC}},m)_{\text{Init}'_{\text{DIC}}}$	
5			$\mathit{send}(\texttt{Send},\texttt{sid}_{\text{SC}},\lvert m\rvert)_{\text{Adv}}$	
6			$\mathit{receive}(\texttt{Send},\texttt{sid}_{\text{SC}},t)_{\text{Rec}'_{\text{DIC}}}$	$\mathit{receive}(\texttt{Send},\texttt{sid}_{\text{SC}},\lvert m\rvert)_{\text{F}_{\text{SC}}}$
7			$\mathit{send}(\texttt{Send},\texttt{sid}_{\text{SC}},\lvert t\rvert)_{\text{Adv}}$	$\mathit{send}(\texttt{Response},\texttt{sid}_{\text{SC}},ok)_{\text{F}_{\text{SC}}}$
8			$\mathit{receive}(\texttt{Response},\texttt{sid}_{\text{SC}},ok)_{\text{Adv}}$	$\mathit{receive}(\texttt{Send},\texttt{sid}_{\text{SC}},\lvert t\rvert)_{\text{F}_{\text{SC}}}$
9			$\mathit{send}(\texttt{Receive},\texttt{sid}_{\text{SC}},mes)_{\text{Rec}'_{\text{DIC}}}$	$\mathit{send}(\texttt{Response},\texttt{sid}_{\text{SC}},ok)_{\text{F}_{\text{SC}}}$
10		$\mathit{receive}(\texttt{Receive},\texttt{sid}_{\text{SC}},m)_{\text{F}_{\text{SC}}}$	$\mathit{receive}(\texttt{Response},\texttt{sid}_{\text{SC}},ok)_{\text{Adv}}$	
11			$\mathit{send}(\texttt{Receive},\texttt{sid}_{\text{SC}},mes)_{\text{Init}'_{\text{DIC}}}$	
12	$\mathit{receive}(\texttt{Receive},\texttt{sid}_{\text{SC}},m)_{\text{F}_{\text{SC}}}$			
13		$\mathit{out}(\texttt{Receive},\texttt{sid}_{\texttt{DIC}},\mathit{plain})_{\text{Rec}'}$		

Ideal Execution

No.	$\overline{\text{Init}'_{\text{DIC}}}$	$\overline{\text{Rec}'_{\text{DIC}}}$	F_{DIC}	Sim'_{DIC}
1	$\mathit{in}(\texttt{Send},\texttt{sid}_{\texttt{DIC}},m)_{\overline{\text{Init}'_{\text{DIC}}}}$			
2	$\mathit{send}(\texttt{Send},\texttt{sid}_{\texttt{DIC}},\mathit{smes})_{\text{F}_{\text{DIC}}}$		$\mathit{receive}(\texttt{Send},\texttt{sid}_{\texttt{DIC}},m)_{\overline{\text{Init}'_{\text{DIC}}}}$	
3			$\mathit{send}(\texttt{Send},\texttt{sid}_{\texttt{DIC}},m)_{\text{Adv}}$	
4				$\mathit{receive}(\texttt{Send},\texttt{sid}_{\texttt{DIC}},m)_{\text{F}_{\text{DIC}}}$
5				$\mathit{send}(\texttt{Response},\texttt{sid}_{\texttt{DIC}},ok)_{\text{F}_{\text{DIC}}}$
6			$\mathit{receive}(\texttt{Response},\texttt{sid}_{\texttt{DIC}},ok)_{\text{Adv}}$	
7			$\mathit{send}(\texttt{Receive},\texttt{sid}_{\texttt{DIC}},mes)_{\overline{\text{Rec}'_{\text{DIC}}}}$	
8		$\mathit{receive}(\texttt{Send},\texttt{sid}_{\texttt{DIC}},m)_{\text{F}_{\text{DIC}}}$		
9		$\mathit{out}(\texttt{Receive},\texttt{sid}_{\texttt{DIC}},m)_{\overline{\text{Rec}'_{\text{DIC}}}}$		

Table 7.21: Corresponding Task Sequence in Data Sending Session for $\text{Real}'_{\text{DIC}}$ and $\text{Ideal}'_{\text{DIC}}$ under $M_{\pi'_{\text{DIC}}}$

Real Execution

$\mathrm{Init}'_{\mathrm{DIC}}$

	active		*ntask*		*smes*		*rmes*		*dummy*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\bot$	DSS2	$\bot$	*m*	-	-	-	-
DSS2	-	-	DSS2	DSS4	-	$\bot$	-	-	-	-
DSS3	-	-	$\bot$	DSS2	-	*s*	-	-	-	$\top$
DSS4	$\top$	-	DSS4	*	-	-	$\bot$	*r*	-	-
DSS5	-	-	DSS5	$\bot$	-	-	-	$\bot$	-	$\bot$
Initial value	$\top$		$\bot$		$\bot$		$\bot$		$\bot$	
Final value	$\top$		$\bot$		$\bot$		$\bot$		$\bot$	

* If *dummy* = $\top$ then *rmes* := *m* and *ntask* := DSS5.
Else *rmes* and *ntask* := $\bot$.

Ideal Execution

$\overline{\mathrm{Init}'_{\mathrm{DIC}}}$

	active		*ntask*		*smes*		*rmes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\bot$	DSS2	$\bot$	*m*	-	-
DSS2	-	-	DSS2	$\bot$	-	$\bot$	-	-
DSS3	$\top$	-	$\bot$	DSS4	-	-	$\bot$	*m*
DSS4	-	-	DSS4	$\bot$	-	-	-	$\bot$
Initial value	$\top$		$\bot$		$\bot$		$\bot$	
Final value	$\top$		$\bot$		$\bot$		$\bot$	

Table 7.22: State Fallow of Data Sending Session for $\mathrm{Real}'_{\mathrm{DIC}}$ and $\mathrm{Ideal}'_{\mathrm{DIC}}$ (Part I)

Real Execution

$\mathrm{Rec}'_{\mathrm{DIC}}$

	active		*ntask*		*smes*		*rmes*		*dummy*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	*m*	-	-	-	-
DSS2	-	-	DSS2	DSS4	-	$\perp$	-	-	-	-
DSS3	-	-	$\perp$	DSS2	-	*t*	-	-	-	$\top$
DSS4	$\top$	-	DSS4	*	-	-	$\perp$	*r*	-	-
DSS5	-	-	DSS5	$\perp$	-	-	-	$\perp$	-	$\perp$
Initial value	$\top$		$\perp$		$\perp$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\perp$		$\perp$	

* If *dummy* = $\top$ then *rmes* := *m* and *ntask* := DSS5.
Else *rmes* and *ntask* := $\perp$.

Ideal Execution

$\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$

	active		*ntask*		*smes*		*rmes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	*m*	-	-
DSS2	-	-	DSS2	$\perp$	-	$\perp$	-	-
DSS3	$\top$	-	$\perp$	DSS4	-	-	$\perp$	*m*
DSS4	-	-	DSS4	$\perp$	-	-	-	$\perp$
Initial value	$\top$		$\perp$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\perp$	

Table 7.23: State Flow of Data Sending Session for $\mathrm{Real}'_{\mathrm{DIC}}$ and $\mathrm{Ideal}'_{\mathrm{DIC}}$ (Part II)

Real Execution

	F_{SC}								Adv'_{DIC}					
	active		*ntask*		*mes*		$okcond_{Adv}$		*active*		*ntask*		*length*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	m	-	-	$\top$	-	$\perp$	DSS2	-	$\|m\|$
DSS2	-	-	DSS2	DSS3	-	-	$\perp$	-	-	-	DSS2	$\perp$	-	$\perp$
DSS3	-	-	DSS3	DSS4	-	-	-	$\top$	-	-	-	-	-	-
DSS4	-	-	DSS4	$\perp$	-	$\perp$	-	$\perp$	-	-	-	-	-	-
Initial value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	

Ideal Execution

	F_{DIC}								Sim'_{DIC}					
	active		*ntask*		*mes*		$okcond_{Adv}$		*active*		*ntask*		*mes*	
Process ID	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:	pre:	eff:
DSS1	$\top$	-	$\perp$	DSS2	$\perp$	m	-	-	$\top$	-	$\perp$	DSS2	-	m
DSS2	-	-	DSS2	DSS3	-	-	$\perp$	-	-	-	DSS2	$\perp$	-	$\perp$
DSS3	-	-	DSS3	DSS4	-	-	-	$\top$	-	-	-	-	-	-
DSS4	-	-	DSS4	$\perp$	-	$\perp$	-	$\perp$	-	-	-	-	-	-
Initial value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	
Final value	$\top$		$\perp$		$\perp$		$\perp$		$\top$		$\perp$		$\perp$	

Table 7.24: State Flow of Data Sending Session for $Real'_{DIC}$ and $Ideal'_{DIC}$ (Part III)

Code for Initiator of Direction-Indeterminable Channel, $\text{Init}'_{\text{DIC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\text{Init}', \text{Rec}'\}, \mathtt{sid}'_{\mathtt{DIC}})$
$\mathtt{sid}_{\text{SC}} = (\text{Init}', \text{Rec}', \mathtt{sid}'_{\text{SC}})$

Input:	Output:
$in(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\text{Init}'}$	$send(\mathtt{Establish}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
$in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\text{Init}'}$	$send(\mathtt{Send}, \mathtt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
$receive(\mathtt{Receive}, \mathtt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$	$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, plain)_{\text{Init}'}$
$rand(s)_{sval}$	
$in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\text{Init}'}$	$send(\mathtt{Expire}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$

State:

$smes, rmes \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$active \in \{\perp, \top\}$, initially $\perp$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$dummy \in \{\perp, \top\}$, initially $\perp$

Tasks:

$\{send(\mathtt{Establish}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}, send(\mathtt{Send}, \mathtt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}},$
$out(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, plain)_{\text{Init}'}, send(\mathtt{Expire}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}\}$

Figure 7.35: Code for Initiator of Direction-Indeterminable Channel, $\text{Init}'_{\text{DIC}}$ (Part I)

Code for Initiator of Direction-Indeterminable Channel, $\text{Init}'_{\text{DIC}}$

Transitions:

```
Establish Session:
    ESS1. $in(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{Init}'}$
        pre: $active$ and $ntask = \bot$
        eff: $ntask := \text{ESS2}$
    ESS2. $send(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
        pre: $ntask = \text{ESS2}$
        eff: $active := \top$ and $ntask := \bot$
Data Sending Session:
    DSS1. $in(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}'}$
        pre: $active = \top$, $smes$ and $ntask = \bot$
        eff: $smes := m$ and $ntask := \text{DSS2}$
    DSS2. $send(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
        pre: $m := smes$ and $ntask = \text{DSS2}$
        eff: $smes := \bot$ and $ntask := \text{DSS4}$
    DSS3. $rand(s)_{sval}$
        pre: $ntask = \bot$
        eff: $dummy := \top$, $smes := s$ and $ntask := \text{DSS2}$
    DSS4. $receive(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
        pre: $active = \top$ and $ntask = \text{DSS4}$
        eff: If $dummy = \top$ then $rmes := m$ and $ntask := \text{DSS5}$.
        Else $rmes$ and $ntask := \bot$.
    DSS5. $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, plain)_{\text{Init}'}$
        pre: $plain := rmes$ and $ntask = \text{DSS5}$
        eff: $dummy$, $rmes$ and $ntask := \bot$
```

Figure 7.36: Code for Initiator of Direction-Indeterminable Channel, $\text{Init}'_{\text{DIC}}$ (Part II)

```
Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}'_{\mathrm{DIC}}$

Transitions:

    Expire Session:
        EXS1. $in(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}'}$
            pre: $active = \top$ and $smes, rmes$ and $ntask = \bot$
            eff: $ntask :=$ EXS2
        EXS2. $send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F_{SC}}}$
            pre: $ntask =$ EXS2
            eff: $active$ and $ntask := \bot$
```

Figure 7.37: Code for Initiator of Direction-Indeterminable Channel, $\mathrm{Init}'_{\mathrm{DIC}}$ (Part III)

Code for Receiver of Direction-Indeterminable Channel, $\mathrm{Rec}'_{\mathrm{DIC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}', \mathrm{Rec}'\}, \mathtt{sid}'_{\mathtt{DIC}})$ $\mathtt{sid}_{\mathrm{SC}} = (\mathrm{Init}', \mathrm{Rec}', \mathtt{sid}'_{\mathrm{SC}})$

Input:	Output:
$\mathit{in}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}'}$	$\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
$\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}'}$	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$
$\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$	$\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{plain})_{\mathrm{Rec}'}$
$\mathit{rand}(t)_{\mathit{tval}}$	
$\mathit{in}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}'}$	$\mathit{send}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$

State:

$\mathit{smes}, \mathit{rmes} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$\mathit{active} \in \{\perp, \top\}$, initially $\perp$
$\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$\mathit{dummy} \in \{\perp, \top\}$, initially $\perp$

Tasks:

$\{\mathit{send}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}, \mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}},$
$\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{plain})_{\mathrm{Rec}'}, \mathit{send}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}\}$

Figure 7.38: Code for Receiver of Direction Indeterminable Channel, $\mathrm{Rec}'_{\mathrm{DIC}}$ (Part I)

Code for Receiver of Direction-Indeterminable Channel, Rec'_{DIC}

Transitions:

Establish Session:

```
ESS1. $in(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{Rec}'}$
        pre: $active$ and $ntask = \perp$
        eff: $ntask :=$ ESS2
ESS2. $send(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
        pre: $ntask =$ ESS2
        eff: $active := \top$ and $ntask := \perp$
```

Data Sending Session:

```
DSS1. $in(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Rec}'}$
        pre: $active = \top$, $smes$ and $ntask = \perp$
        eff: $smes := m$ and $ntask :=$ DSS2
DSS2. $send(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
        pre: $m := smes$ and $ntask =$ DSS2
        eff: $smes := \perp$ and $ntask :=$ DSS4
DSS3. $rand(t)_{tval}$
        pre: $ntask = \perp$
        eff: $dummy := \top$, $smes := t$ and $ntask :=$ DSS2
DSS4. $receive(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
        pre: $active = \top$ and $ntask =$ DSS4
        eff: if $dummy = \top$ then $rmes := m$ and $ntask :=$ DSS5
        else $rmes := \perp$ and $ntask := \perp$
DSS5. $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, plain)_{\text{Rec}'}$
        pre: $plain := rmes$ and $ntask =$ DSS5
        eff: $dummy$, $rmes$ and $ntask := \perp$
```

Figure 7.39: Code for Receiver of Direction Indeterminable Channel, Rec'_{DIC} (Part II)

```
Code for Receiver of Direction-Indeterminable Channel, $\mathrm{Rec}'_{\mathrm{DIC}}$

Transitions:

    Expire Session:
        EXS1. $\mathit{in}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}'}$
              pre: $\mathit{active} = \top$ and $\mathit{smes}, \mathit{rmes}$ and $\mathit{ntask} = \bot$
              eff: $\mathit{ntask} := \mathrm{EXS2}$
        EXS2. $\mathit{send}(\mathtt{Expire}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
              pre: $\mathit{ntask} = \mathrm{EXS2}$
              eff: $\mathit{active}$ and $\mathit{ntask} := \bot$
```

Figure 7.40: Code for Receiver of Direction Indeterminable Channel, $\mathrm{Rec}'_{\mathrm{DIC}}$ (Part III)

Code for Adversary for Secure Channel, $\mathrm{Adv}'_{\mathrm{DIC}}$

Signature:

$\mathtt{sid}_{\mathrm{SC}} = (\mathrm{Init}, \mathrm{Rec}, \mathtt{sid}'_{\mathrm{SC}})$

Input:
$\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F_{SC}}}$
$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{F_{SC}}}$
$\mathit{receive}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F_{SC}}}$

Output:
$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, \mathit{ok})_{\mathrm{F_{SC}}}$

Other:
*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message and $\mathit{out}(*)$.

State:

$\mathit{active} \in \{\bot, \top\}$, initially $\bot$
$\mathit{ntask} \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$
$\mathit{length} \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Tasks:

$\{\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, \mathit{ok})_{\mathrm{F_{SC}}}$, other arbitrary tasks$\}$

Figure 7.41: Code for Adversary for Secure Channel, $\mathrm{Adv}'_{\mathrm{DIC}}$ (Part I)

Code for Adversary for Secure Channel, $\mathrm{Adv}'_{\mathrm{DIC}}$

Transitions:

```
    Establish Session:
        ESS1. $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$
            pre: $active = \bot$
            eff:$active := \top$
    Data Sending Session:
        DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, |m|)_{\mathrm{F_{SC}}}$
            pre: $active = \top$ and $ntask = \bot$
            eff: $length := |m|$ and $ntask := \mathrm{DSS2}$
        DSS2. $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{SC}}, ok)_{\mathrm{F_{SC}}}$
            pre: $ntask = \mathrm{DSS2}$
            eff: $length := \bot$ and $ntask := \bot$
    Expire Session:
        EXS1. $receive(\mathtt{Expire}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$
            pre: $active = \top$
            eff: $active := \bot$
    Other tasks:
        This adversary makes other arbitary tasks.
```

Figure 7.42: Code for Adversary for Secure Channel, $\mathrm{Adv}'_{\mathrm{DIC}}$ (Part II)

Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}'_{\mathrm{DIC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\overline{\mathrm{Init}}, \overline{\mathrm{Rec}}\}, \mathtt{sid}'_{\mathtt{DIC}})$

Input:
$receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F_{DIC}}}$
$receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$

Output:
$send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F_{DIC}}}$

Other:
*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message and $out(*)$.

State:

$active \in \{\perp, \top\}$, initially $\perp$ $\quad$ $mes \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$

Other arbitrary variables; call ”new” variables.

Tasks:

$\{send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F_{DIC}}}\ \}$

Figure 7.43: Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}'_{\mathrm{DIC}}$ (Part I)

Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}'_{\mathrm{DIC}}$

Transitions:

Establish Session:

```
ESS1. $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F_{DIC}}}$
        pre: $active$ and $ntask = \perp$
        eff:$active := \top$
```

Data Sending Session:

```
DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F_{DIC}}}$
        pre: $active = \top$ and $ntask = \perp$
        eff: $mes := m$ and $ntask :=$ DSS2
DSS2. $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F_{DIC}}}$
        pre: $ntask =$ DSS2
        eff: $mes := \perp$ and $ntask := \perp$
```

Expire Session:

```
EXS1. $receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F_{DIC}}}$
        pre: $active = \top$   eff: $active := \perp$
```

Other tasks:

This simulator makes arbitrary tasks to simulate the real world protocol system $\mathrm{Real}'_{\mathrm{DIC}}$. The tasks mey be run with the information obtained from the simulator. Additionaly, this simulator can output the message from the adversary in the simiulating world to the environment.

Figure 7.44: Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}'_{\mathrm{DIC}}$ (Part II)

7.3.2 Reduction of SC to DIC

Let n be the number of parties and $M_{rasync}(t_1^*, \cdots, t_n^*)$ be master schedules where t_i^* is a task in party P_i.

Definition 26. **[$M_{rasync}(t_1^*, \cdots, t_n^*, k)$]** *Let k be a integer. Let t_i^* be a task specified by ρ_i for party P_i. Let c_i be the number of times t_i^* is scheduled by M. M schedules the task activations of $t_1^*, \cdots, t_n^*$ so that $|c_i - c_j| \leq k$ for all i, j in a random order.*

Note that we must consider a property similar to the Chernov bound property if we employ this master schedule to use the key exchange among parties for safe exchange.

Let π_{SC} be a protocol of SC. Let $M_{\pi_{\mathrm{SC}}}$ and $M'_{\pi_{\mathrm{SC}}}$ be $M_{psync}(\mathrm{Init}_{\mathrm{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F_{DIC}}}, \mathrm{Rec}_{\mathrm{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}})$ and $M_{rasync}(\mathrm{Init}_{\mathrm{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F_{DIC}}}, \mathrm{Rec}_{\mathrm{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}}, k)$, respectively.

$M_{\pi_{\mathrm{SC}}}$ and $M'_{\pi_{\mathrm{SC}}}$ are accepted for the master schedule, hereafter, we explain by using $M_{\pi_{\mathrm{SC}}}$.

Let $\mathrm{Init}_{\mathrm{SC}}$ and $\mathrm{Rec}_{\mathrm{SC}}$ be the initiator code and receiver code for a real system, see Fig.7.45, Fig.7.46 and Fig.7.47, Fig.7.48, Fig.7.49 and Fig.7.50, respectively. Let $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$ be the initiator code and receiver code for an ideal system, see Fig.7.53 and Fig.7.54, and Fig.7.55 and Fig.7.56, respectively. Finally, let $\mathrm{Adv}_{\mathrm{SC}}$, $\mathrm{Sim}_{\mathrm{SC}}$, and $\mathrm{F_{SRC}}$ be the adversary code, the simulator code, and the random bit generator code in Fig.7.51 and Fig.7.52, and Fig.7.57 and Fig.7.58, and Fig.7.59, respectively. Let $\mathrm{Real}_{\mathrm{SC}}$ and $\mathrm{Ideal}_{\mathrm{SC}}$ be a SC protocol system and a SC functionality system, respectively, defined as follows:

$$\mathrm{Real}_{\mathrm{SC}} := hide(\mathrm{Init}_{\mathrm{SC}} || \mathrm{Rec}_{\mathrm{SC}} || \mathrm{Adv}_{\mathrm{SC}} || \mathrm{F_{SRC}} || \mathrm{F_{DIC}}, \{rand(*)\}),$$
$$\mathrm{Ideal}_{\mathrm{SC}} := hide(\overline{\mathrm{Init}_{\mathrm{SC}}} || \overline{\mathrm{Rec}_{\mathrm{SC}}} || \mathrm{Sim}_{\mathrm{SC}} || \mathrm{F_{SC}}, \{rand(*)\}).$$

Tasks $\overline{\mathrm{Init}_{\mathrm{SC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{SC}}}$ relay input messages from the environment to the ideal functionality task and relay received messages from the ideal functionality task to the environment, respectively, as interface parties in the ideal system.

Theorem 7. *SC protocol system* $\mathrm{Real}_{\mathrm{SC}}$ *perfectly hybrid-implements SC functionality system* $\mathrm{Ideal}_{\mathrm{SC}}$ *with respect to an adaptive adversary under master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F_{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}})$ *(SC is reducible to DIC with respect to an adaptive adversary under master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}} s)_{\mathrm{F_{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}})$*).*

The proof of theorem 7 is described similarly to other theorems. The master schedule can be M_{rasync} instead of M_{psync}.

Let ϵ_{R} and ϵ_{I} be discrete probability measures on finite executions of $\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}$ and $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$, respectively. We prove Theorem 7 by showing that ϵ_{R} and ϵ_{I} satisfy the trace distribution property, $tdist(\epsilon_{\mathrm{R}}) = tdist(\epsilon_{\mathrm{I}})$. Here, we define correspondence R between the states in $\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}$ and the states in $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$. We say $(\epsilon_{\mathrm{R}}, \epsilon_{\mathrm{I}}) \in R$ if and only if for every $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$, all state correspondences in Tables 7.25, 7.26 and 7.27 hold. We then prove R is a simulation relation in Lemma 4.

Lemma 4. *Relation R defined above is a simulation relation from* $\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}$ *to* $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$ *under master schedule* $M_{\pi_{\mathrm{SC}}}$.

Proof. We prove that R is a simulation relation from $\mathrm{Real}_{\mathrm{SC}}|\mathrm{Env}$ to $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$ using mapping corrtask $R^{*}_{\mathrm{Real_{SC}|Env}} \times R_{\mathrm{Real_{SC}|Env}} \rightarrow R^{*}_{\mathrm{Ideal_{SC}||Env}}$, which is defined as follows.

The task sequence of system $\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}$ are perfectly correspond to the task sequence of system $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$ under schedule $M_{\pi_{\mathrm{SC}}}$. Formally, to prove that R is a simulation relation from $\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}$ to $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$, we show that R satisfies the start condition and step condition.

- **Start condition**
  It is true that the start states of s and u in $\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}$ and $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$, respectively, are on the Dirac measures. That is, the start states of s and u satisfy relation R because the start states of s and u are all $\perp$ for each task on master schedule $M_{\pi_{\mathrm{SC}}}$. Therefore, the trace distribution property holds.

- **Step condition**
  Let $\epsilon'_{\mathrm{R}} = apply(\epsilon_{\mathrm{R}}, T)$ and $\epsilon'_{\mathrm{I}} = apply(\epsilon_{\mathrm{I}}, corrtasks(\rho, T))$. If $(\epsilon_{\mathrm{R}}, \epsilon_{\mathrm{I}}) \in$

R, $\rho \in R^*_{\mathrm{Real_{SC}}||\mathrm{Env}}$, ϵ_{R} is consistent with ρ, then ϵ_{I} is consistent with $full(corrtasks)(\rho)$, and $T \in \mathrm{Real_{SC}}||\mathrm{Env}$. Then there exist the following.

- Probability measure p on countable index set I,
- Probability measures $\epsilon'_{\mathrm{R},j}$, $j \in I$, on finite executions of $\mathrm{Real_{SC}}||\mathrm{Env}$, and
- Probability measures $\epsilon'_{\mathrm{I},j}$, $j \in I$, on finite executions of $\mathrm{Ideal_{SC}}||\mathrm{Env}$,

such that:

- For each $j \in I$, $\epsilon'_{\mathrm{R},j}\ R\ \epsilon'_{\mathrm{I},j}$,
- $\Sigma_{j\in I}\, p(j)(\epsilon'_{\mathrm{R},j}) = apply(\epsilon_{\mathrm{R}}, T)$, and
- $\Sigma_{j\in I}\, p(j)(\epsilon'_{\mathrm{I},j}) = apply(\epsilon_{\mathrm{I}}, corrtask(\rho, T))$.

Task Correspondence

For any $(\rho, T) \in (R^*_{\mathrm{Real_{SC}}||\mathrm{Env}} \times R_{\mathrm{Real_{SC}}||\mathrm{Env}})$, the following task Correspondence, which is also summarized in Table 7.28, holds.

1. **Establish Session**

   (a) $\mathrm{Init_{SC}}.send(\mathtt{Establish_{DIC}}, \mathtt{sid_{DIC}})_{\mathrm{F_{DIC}}}$
   $=_{\mathrm{corr.}}\ \overline{\mathrm{Init_{SC}}}.send(\mathtt{Establish_{SC}}, \mathtt{sid_{SC}})_{\mathrm{F_{SC}}}$
   Let $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ be $send(\mathtt{Establish_{DIC}}, \mathtt{sid_{DIC}})_{\mathrm{F_{DIC}}}$ and $send(\mathtt{Establish_{SC}}, \mathtt{sid_{SC}})_{\mathrm{F_{SC}}}$, respectively. Here, we must consider the cases of $\mathrm{Init_{SC}}$ and $\overline{\mathrm{Init_{SC}}}$, but these follow the same discussion. So, we consider the case of $\overline{\mathrm{Init_{SC}}}$. We assume that for each state, $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. The precondition of $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ is $ntask = \mathrm{ESS2}$ from each codes. $\mathtt{T_{REAL}}$ (resp., $\mathtt{T_{IDEAL}}$) is enabled (or disabled) in s (resp., u) if and only if $s.\mathrm{Init_{SC}}.ntask = \mathrm{ESS2}$ (resp. $u.\overline{\mathrm{Init_{SC}}}.ntask = \mathrm{ESS2}$).

From (j) in Table 7.25, $u.\overline{\text{Init}_{\text{SC}}}.ntask$ and $s.\text{Init}_{\text{SC}}.ntask$ imply that $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ are uniformly enabled or disabled in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

i. Disable Case:
   Let I and p be the set that has a single element and Dirac measure on I, respectively. Let $\epsilon'_{\text{R},1} = \epsilon'_{\text{R}}$ and $\epsilon'_{\text{I},1} = \epsilon'_{\text{I}}$. We have the fact that $\epsilon'_{\text{R}} = \epsilon_{\text{R}}$ and $\epsilon'_{\text{I}} = \epsilon_{\text{I}}$. Here, we obtain $\epsilon'_{\text{R},1} R \epsilon'_{\text{I},1}$ from relation $\epsilon_{\text{R}} R \epsilon_{\text{I}}$. The trace distribution equivalence property, $tdist(\epsilon'_{\text{R}}) = tdist(\epsilon'_{\text{I}})$, also holds since $tdist(\epsilon_{\text{R}}) = tdist(\epsilon_{\text{I}})$ under $M_{\pi_{\text{SC}}}$.

ii. Enable Case:
   Let q denote the state of precondition : $ntask$ = ESS2. Let $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ be the action enabled in q in each world. We show that each of $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ is a unique action that is enabled in q. From the definition of $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$, the precondition is only $ntask$ = ESS2, and is unique in all tasks in Init_{SC} and $\overline{\text{Init}_{\text{SC}}}$. Then, there are two unique effects that update the $active$ and $ntask$ to be $\top$ and $\perp$, respectively. From the precondition and the effect of $\mathtt{T_{REAL}}$, and the state equivalence of (i) and (j), we obtain that the subsequent action of $\mathtt{T_{REAL}}$ (and $\mathtt{T_{IDEAL}}$) is also a unique action that is enabled in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).
   Let I and p be the set that has a single element and the Dirac measure on I, respectively. Let $\epsilon'_{\text{R},1} = \epsilon'_{\text{R}}$ and $\epsilon'_{\text{I},1} = \epsilon'_{\text{I}}$. Here, we establish the property of R for ϵ'_{R} and ϵ'_{I} to show that $(\epsilon'_{\text{R}}, \epsilon'_{\text{I}}) \in R$. Then we show trace distribution equivalence for ϵ'_{R} and ϵ'_{I}. To establish this property, we consider any state $s' \in$ supp.lst(ϵ'_{R}) and $u' \in$ supp.lst(ϵ'_{I}). Let s be any state in supp.lst(ϵ_{R}) such that $s' \in$ supp(μ_{s}) where $(s, \zeta, \mu_s) \in$ $\text{Real}_{\text{SC}}||\text{Env}$. Let u be any state in supp.lst(ϵ_{I}) such that $u' \in$ supp(μ_{u}) where $(u, corrtask(\rho, \zeta), \mu_u) \in \text{Ideal}_{\text{SC}}||\text{Env}$.
   It is true that $\mathtt{T_{REAL}}$ updates Init.$active$ to $\top$ and

$\mathrm{Init}_{\mathrm{SC}}.\mathit{ntask}$ to $\perp$ from the definition of the effect of $\mathtt{T}_{\mathtt{REAL}}$. Similarly, $\mathtt{T}_{\mathtt{IDEAL}}$ updates $\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{active}$ to $\top$ and $\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{ntask}$ to $\perp$ from the definition of the effect of $\mathtt{T}_{\mathtt{IDEAL}}$. From the state equivalence of (i) and (j) in Table 7.25, we have $u.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{active} = s.\mathrm{Init}_{\mathrm{SC}}.\mathit{active}$ and $u.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{ntask} = s.\mathrm{Init}_{\mathrm{SC}}.\mathit{ntask}$. We obtain that $u'.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{active} = s'.\mathrm{Init}_{\mathrm{SC}}.\mathit{active}$ and $u'.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{ntask} = s'.\mathrm{Init}_{\mathrm{SC}}.\mathit{ntask}$. By the definition of $\mathrm{Init}_{\mathrm{SC}}$ and $\overline{\mathrm{Init}_{\mathrm{SC}}}$, $\mathtt{T}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is a unique action that updates the state of *active* of $\mathrm{Real}_{\mathrm{SC}}$ (resp., $\mathrm{Ideal}_{\mathrm{SC}}$). Therefore, we obtain the trace distribution property, $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

(b) $\mathrm{Rec}_{\mathrm{SC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F_{DIC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{SC}}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F_{SC}}}$

This is similar to case 1a. The precondition and effect of these tasks are identical to each other. The preconditions of the task on the left side of the equation are $\mathit{active} = \top$ and $\mathit{ntask} = \mathrm{ESS2}$. This is equivalent to those on the right side of the equation. The effect of the task on left side is $\mathit{ntask} := \perp$. This effect is also the same as that on the right side. Let $\mathtt{T}_{\mathtt{REAL}}$ be $\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$. Let $\mathtt{T}_{\mathtt{IDEAL}}$ be $\mathrm{F}_{\mathrm{SC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}}$. We show that $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. We consider that for each state $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. Then, $\mathtt{T}_{\mathtt{REAL}}$ is enabled (or disabled) in s if and only if $s.\mathrm{T}_{\mathtt{REAL}}.\mathit{active} = \top$ and $s.\mathrm{T}_{\mathtt{REAL}}.\mathit{ntask} = \mathrm{ESS2}$. The precondition of $\mathtt{T}_{\mathtt{IDEAL}}$, ($n$) in the Table 7.25, implies that $\mathtt{T}_{\mathtt{IDEAL}}$ is uniformly enabled or disabled. The rest of this proof is similar to that for the task of the initiator.

More specifically, the precondition and effect of the real task are the same as those for the ideal task from (n), and (m) and (n), respectively. So, these tasks correspond.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definitions and the state correspondence of pre: (n). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondence of pre: (n), and state correspondences of eff: (m) and (n), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(c) $\mathrm{F_{DIC}}.send(\mathtt{SID}, \mathtt{sid_{DIC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F_{SC}}.send(\mathtt{SID}, \mathtt{sid_{SC}})_{\mathrm{Adv}}$

The precondition and effect of these tasks are identical to each other. The preconditions of the task on the left side of the equation are $active = \top$ and $ntask = \mathrm{ESS2}$. These are equivalent to those for the right side of the equation . The effect of the task on the left side of the equation is $ntask := \perp$. This effect is also the same as that for the right side. Let $\mathtt{T_{REAL}}$ be $\mathrm{F_{DIC}}.send(\mathtt{SID}, \mathtt{sid_{DIC}})_{\mathrm{Adv}}$. Let $\mathtt{T_{IDEAL}}$ be $\mathrm{F_{SC}}.send(\mathtt{SID}, \mathtt{sid_{SC}})_{\mathrm{Adv}}$. We show that $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. We consider that for each state in $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. Then, $\mathtt{T_{REAL}}$ is enabled (or disabled) in s if and only if $s.\mathtt{T_{REAL}}.active = \top$ and $s.\mathtt{T_{REAL}}.ntask = \mathrm{ESS2}$. The preconditions of $\mathtt{T_{IDEAL}}$, (d) and (f) in Table 7.25, implies that $\mathtt{T_{IDEAL}}$ is uniformly enabled or disabled. The rest of this proof is similar to that for the task of the Initiator.

2. **Data Sending Session**

Here, we can consider the following two cases in this session. One is that Env inputs $in(\mathtt{Send}, \mathtt{sid_{SC}}, m)_{\mathrm{Init}}$ to $\mathrm{Init_{SC}}$ and message receiver $\mathrm{Rec}_{\mathrm{SC}}$ outputs message $out(\texttt{Receive}, \texttt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Rec}}$. The other is that Env inputs $in(\texttt{Send}, \texttt{sid}_{\mathrm{SC}}, m)_{\mathrm{Rec}}$ to $\mathrm{Rec}_{\mathrm{SC}}$ and message receiver $\mathrm{Init}_{\mathrm{SC}}$ outputs message $out(\texttt{Receive}, \texttt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Init}}$. These two cases are considered to be the same, so hereafter we consider the first case. The basic task sequences are given in Table 7.29. Note that the simulation has the same sequences as those for the Real Execution in Table 7.29.

The key exchange phase is simulated by $\mathrm{Sim}_{\mathrm{SC}}$, and more details are given in 1. Hereafter, we explain that the message sending is safely executed after the key exchange.

(a) $\mathrm{Init}_{\mathrm{SC}}.send(\texttt{Send}, \texttt{sid}_{\mathrm{DIC}}, cipher)_{\mathrm{F}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{SC}}}.send(\texttt{Send}, \texttt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$
The precondition and effect of these tasks are identical to each other. Let $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ be $send(\texttt{Send}, \texttt{sid}_{\mathrm{DIC}}, cipher)_{\mathrm{F}_{\mathrm{DIC}}}$ and $send(\texttt{Send}, \texttt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$. We show that $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. We consider that for each state in $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. $\mathtt{T}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is enabled (or disabled) if and only if $s.\mathrm{Init}_{\mathrm{SC}}.ntask = \mathrm{DSS2}$ (resp., $u.\overline{\mathrm{Init}_{\mathrm{SC}}}.ntask = \mathrm{DSS2}$). ($g$) and ($j$) in Table 7.25 imply that $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. So, these tasks are activated under the same conditions.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondences of pre: (g) and (j). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondences of pre: (g)

and (j), and state correspondences of eff: (g) and (j), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(b) $\mathrm{Rec}_{\mathrm{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, cipher)_{\mathrm{F_{DIC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{SC}}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}}$

This is identical to case 2a. The states of precondition and effect of both expression are same. The rest of this proof is similar to 2a.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondences of pre: (k) and (n). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definitions, the state correspondences of pre: (k) and (n), and state correspondences of eff: (k) and (n), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(c) $\mathrm{F_{DIC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$
$=_{\mathrm{corr.}} \mathrm{F_{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}}$

The precondition and effect of these tasks are identical to each other. Let $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ be $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}}$ and $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}}$. We show that $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. We consider that for each state in $s \in \mathrm{supp.lst}(\epsilon_{\mathrm{R}})$ and $u \in \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ are fixed. $\mathtt{T}_{\mathtt{REAL}}$ (resp., $\mathtt{T}_{\mathtt{IDEAL}}$) is enabled (or disabled) if and only if $s.\mathtt{T}_{\mathtt{REAL}}.ntask = \mathrm{DSS2}$ (resp., $u.\mathtt{T}_{\mathtt{IDEAL}}.ntask = \mathrm{DSS2}$).

(c), (d) and (f) in Table 7.25 imply that $\mathtt{T}_{\mathtt{REAL}}$ and $\mathtt{T}_{\mathtt{IDEAL}}$ are uniformly enabled or disabled in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}). So, these tasks are activated under the same conditions.

(d) $\mathrm{Adv}_{\mathrm{SC}}.\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{SC}}}$
The precondition and effect for the real task and are the same as those for the ideal task. The precondition is only ntask = DSS2 and the effect is $\mathit{ntask} := \perp$. From (C) in Table 7.26, these tasks are enabled (or disabled) in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

(e) $\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\overline{X}}$
$=_{\mathrm{corr.}}$ $\mathrm{F}_{\mathrm{SC}}.\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{mes})_{\overline{X}}$
The precondition of the task on the left side of the equation , ntask = DSS4, is that for the task on the right side. The effects of the task on the left, $\mathit{okcond}_{\mathrm{Adv}}, \mathit{mes}$ and $\mathit{ntask} := \perp$, are also identical to those for the task on the right side. The rest of this proof is analogous to case 2a.

(f) $\mathrm{Init}_{\mathrm{SC}}.\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{plain})_{\mathrm{Init}}$
$=_{\mathrm{corr.}}$ $\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{mes})_{\overline{\mathrm{Init}}}$
The states of precondition are $\mathit{plain} := \mathit{rmes}$ and ntask = DSS4. Then, the effects of these tasks are the same. So, if (h) and (j) in Table 7.25 hold, then these tasks are enabled (or disabled) in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondences of pre: (h) and (j). Therefore, we obtain $\mathit{trace}(\epsilon'_{\mathrm{R}}) = \mathit{trace}(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondences of pre: (h) and (j), and state correspondences of eff: (h) and (j), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(g) $\mathrm{Rec}_{\mathrm{SC}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Rec}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{SC}}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\overline{\mathrm{Rec}}}$
This is identical to case 2f. The states of precondition and effect of both expressions are the same.

i. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondences of pre: (l) and (n). Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definitions, the state correspondences of pre: (l) and (n), and state correspondences of eff: (l) and (n), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

3. **Expire Session**

(a) $\mathrm{Init}_{\mathrm{SC}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{SC}}}.send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
The states of precondition and effect or $send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ are the same as those for $send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$) where $ntask$ = EXS2. That is, if (j) in Table 7.25 holds, then these tasks are enabled (or disabled) in every state in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}).

i. Disable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondences of pre: (i) and (j). Therefore, we obtain $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

ii. Enable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondence of pre: (j), and state correspondences of eff: (i) and (j), we have that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

(b) $\mathrm{Rec}_\mathrm{SC}.send(\mathtt{Expire}_\mathtt{DIC}, \mathtt{sid}_\mathtt{DIC})_{\mathrm{F_{DIC}}}$
$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_\mathrm{SC}}.send(\mathtt{Expire}_\mathtt{DIC}, \mathtt{sid}_\mathrm{SC})_{\mathrm{F_{SC}}}$
This case is analogous to the above case 3a. The precondition and effect for the real task are the same as those for the ideal task. The precondition is only $ntask = \mathrm{EXS2}$ and the effects are $active := \bot$ and $ntask := \bot$From (n) in Table 7.25 these tasks are enabled (or disabled) in every state in $\mathrm{supp.lst}(\epsilon_\mathrm{R}) \cup \mathrm{supp.lst}(\epsilon_\mathrm{I})$.

i. Disable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondences of pre: (m) and (n). Therefore, we obtain $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

ii. Enable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondence of pre: (n), and state correspondences of eff: (m) and (n), we have that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain

$trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

(c) $\mathrm{F}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$
$=_{\mathrm{corr.}} \mathrm{F}_{\mathrm{SC}}.send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$
The precondition and effect for the real task are the same as those for the ideal task. The precondition is only $ntask = \mathrm{EXS2}$ and the effects are $active := \bot$ and $estcond_X := \bot$ for all X (and $estcond_{\mathrm{Init}}, estcond_{\mathrm{Rec}} := \bot$ in F_{SC}) and $ntask := \bot$. From (a), (b), (d), and (f) in Table 7.25 these tasks are enabled (or disabled) in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. The rest of this proof is analogous to (a) in the establish session.

Environment Env

From the task definitions and state correspondence (o) in Table7.25, the provability measures for both tasks are uniformly enabled or disabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

Claim 1 The state of Env remains static in all states in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. Let q_e denote this state of Env. This follows from state correspondence o.

Claim 2 If T is a task of Env, then T is either enabled or disabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$ (simultaneously). Furthermore, if T is enabled in all states in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$, then:

1. There exists unique action $a \in \mathrm{T}$ that is enabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.
2. There exists a unique transition of Env from q_e with action a. Let $tr_e = (q_e, a, \mu_e)$ be this transition.

By considering Claim 7.3.2, task T of Env is uniformly enabled or disabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. If T is disabled, let $I = 1$, we obtain $\epsilon'_{\mathrm{R},1} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I},1} = \epsilon_{\mathrm{I}}$, and this results in that $\epsilon'_{\mathrm{R},1} R \epsilon'_{\mathrm{I},1}$ since we have $\epsilon_{\mathrm{R}} R \epsilon_{\mathrm{I}}$. If T is enabled in in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$, Claim 7.3.2 implies that there exists unique action a in every state in supp.lst$(\epsilon_{\mathrm{R}}) \cup$ supp.lst(ϵ_{I}) and transition tr_e of Env from q_e enabled with action a, where $tr_e = (q_e, a, \mu_e)$.

Non Corrupted Case:

1. a is an input / output action of Init. We assume that a is an input action such as $in(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Init}}$, $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Init}}$, $in(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Init}}$, and $out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Init}}$.

   Let s be any state such that $s' \in \mathrm{supp}(\mu_{\mathrm{s}})$, where $(s, a, \mu_s) \in D_{\mathrm{Real_{SC}}||\mathrm{Env}}$. Let u be any state such that $u' \in \mathrm{supp}(\mu_{\mathrm{u}})$, where $(u, a, \mu_u) \in D_{\mathrm{Ideal_{SC}}||\mathrm{Env}}$. For each a, we check that the state correspondences for s' and u' hold if those for s and u hold. If each a is input from Env, then the precondition and effect for the real task are exactly the same as those for the ideal task. For example, if the input message is $in(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Init}}$, then the precondition is $active, ntask = \perp$ and the effect is $ntask :=$ ESS2. These states for the real task correspond to those for the ideal task. So, in the case that the task is enabled (or disabled), the state correspondences of (o), (i), and (j) for s' and u' hold, if the state correspondencs for s and u hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$. This result also works well in the case of $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Init}}$ and $in(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Init}}$.

2. a is an input / output action of Rec. We assume that a is an input action such as $in(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Rec}}$, $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Rec}}$, $in(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Rec}}$ and $out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Rec}}$. This is analogous to 1.

3. a is an input action of Adv. This means that $a = input(g)_{\mathrm{Adv}}$ for some fixed g. For example, g is a corrupt message for some $party \in \{\mathrm{Init}, \mathrm{Rec}\}$. From the fact that the state correspondences $(A) \sim (V)$ for s and u hold, we obtain that the state correspondences for s'

and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

4. a is an internal or an output action of Env. Task a for the real world is identical to that for the ideal world. From the fact that the state correspondence of (o) for s and u hold, we obtain that the state correspondence of (o) for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R},j}) = trace(\epsilon'_{\mathrm{I},j})$.

Corrupted Case:

1. a is an input action of Adv and $party \in \{\mathrm{Init}, \mathrm{Rec}\}$ Here, the party is included in the case of $\mathrm{Init} \wedge \mathrm{Rec}$. Let q_{Adv} be the state of Adv or Sim that is the same in all $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. Let $tr_{\mathrm{Adv}} = (q_{\mathrm{Adv}}, a, \mu_{\mathrm{Adv}})$ be a transition of Adv with action a from q_{Adv}. From Claim 7.3.2, tr_{Adv} is a unique transition. Here, we suppose that $\mathrm{supp}((\mu_{\mathrm{e}} \times \mu_{\mathrm{Adv}}))$ is the pair set $\{(q_{1,j}, q_{2,j}) : j \in I\}$, where I is a countable set. Let p be the probability measures such that for each j, $p(j) = (\mu_e \times \mu_{\mathrm{Adv}})(q_{1,j}, q_{2,j})$. For each j, let $\epsilon'_{\mathrm{R},j}$ be $\epsilon'_{1,j}(\alpha) = \epsilon_1(\alpha')$ where $\alpha \in \mathrm{supp}(\epsilon'_1)$ such that $\mathtt{lst}(\alpha).\mathrm{Env} = q_{1,j}$ and $\mathtt{lst}(\alpha).\mathrm{Adv} = q_{2,j}$. The $\epsilon'_{2,j}$ is analogously constructed from ϵ'_2.

   The rest of this proof is he same as that for 1 by considering state correspondence in each case $party \in \{\mathrm{Init}, \mathrm{Rec}, \mathrm{Init} \wedge \mathrm{Rec}\}$. Finally, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R},j}) = trace(\epsilon'_{\mathrm{I},j})$.

Adversary Adv

From the task definitions and state correspondences $(A) \sim (V)$ in Table 7.26, the provability measures for both tasks are uniformly enabled or disabled in every state in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$.

Claim 3 The state of Adv or Sim is the same in all states in $\mathrm{supp.lst}(\epsilon_{\mathrm{R}}) \cup \mathrm{supp.lst}(\epsilon_{\mathrm{I}})$. Let q_{Adv} denote this state of Adv and Sim. This follows from state correspondence of Sim.

Claim 4 If T is a task of Adv, then T is either enabled or disabled in every state in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I). Furthermore, if T is enabled in all states in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I), then:

1. There is unique action $a \in$ T that is enabled in every state in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I).
2. There is a unique transition of Adv from q_{Adv} with action a, and let $tr_{\text{Adv}} = (q_{\text{Adv}}, a, \mu_{\text{Adv}})$ be this transition.

By considering Claim.7.3.2, task T of Adv is uniformly enabled or disabled in every state in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I). If T is disabled, let $I = 1$, we obtain $\epsilon'_{R,1} = \epsilon_R$ and $\epsilon'_{I,1} = \epsilon_I$, and the result is $\epsilon'_{R,1} R \epsilon'_{I,1}$ since we have $\epsilon_R R \epsilon_I$. If T is enabled, T is enabled in every state in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I). Claim 7.3.2 implies that there is unique action a in every state in supp.lst(ϵ_R) $\cup$ supp.lst(ϵ_I) and transition tr of Adv from q_e enabled with action a, where $tr_{\text{Adv}} = (q_{\text{Adv}}, a, \mu_{\text{Adv}})$. The following cases for the "Non Corrupted Case" and "Corrupted Case" can be considered.

Non Corrupted Case:

1. a is an input action of Env. From the fact that state correspondences $(A) \sim (V)$ for s and u hold, we obtain that state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_R) = trace(\epsilon'_I)$.

2. a is an input or output action of functionality task. This case concerns the message $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\text{F}_{\text{DIC}}}$, $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\text{F}_{\text{DIC}}}$, $receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\text{F}_{\text{DIC}}}$ and $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\text{F}_{\text{DIC}}}$.

   . The rest of this proof is analogous to 1. From the fact that state correspondences $(A) \sim (V)$ for s and u hold, we obtain that state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_R) = trace(\epsilon'_I)$.

3. a is either an output action of Adv that is not an input action of Env, Init, Rec, functionality task, or is an internal action of Adv. This

case concerns "new" tasks. The rest of this proof is analogous to 1. From the fact that state correspondences $(A) \sim (V)$ for s and u hold, we obtain that the state correspondences for s' and u' hold. Therefore, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

4. a is an output action of $out(*)_{adv}$. This case is also works well although this action may effect Env. However, the transition of Env $tr_e = (q_e, a, \mu_e)$ is unique from Claim 7.3.2. Claim 7.3.2 also says that the state of Env remains static in all states in supp.lst(ϵ_{R}) $\cup$ supp.lst(ϵ_{I}). This follows from state correspondence o. Similarly, from the definition and some claims, we obtain the trace distribution property, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Corrupted Case:

This is the case that the static and adaptive adversary Adv corrupt $party \in \{\mathrm{Init}, \mathrm{Rec}\}$.

1. a is an input/output action $in(*)_{party}$, $out(*)_{party}$ of corrupted party, $party \in \{\mathrm{Init}, \mathrm{Rec}\}$. This case is also works well from Claim 7.3.2 and state correspondences in Table 7.25 $\sim$ 7.27.

Perfect Simulation

The simulation of $\mathrm{Sim}_{\mathrm{SC}}$ is perfectly executed for establish session, data sending session, and expire session with respect to no corruption, static corruption and adaptive corruption by an adversary.

1. **No Corruption**

   (a) **Establish Session** First, in the establish session, environment Env sends establish message $in(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\overline{\mathrm{Init}}}$ and message $in(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\overline{\mathrm{Rec}}}$ to initiator $\overline{\mathrm{Init}_{\mathrm{SC}}}$ and receiver $\overline{\mathrm{Rec}_{\mathrm{SC}}}$, respectively. They send establish session messages $send(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$ to F_{SC}. The functionality sends $send(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$ to $\mathrm{Sim}_{\mathrm{SC}}$. After $\mathrm{Sim}_{\mathrm{SC}}$ receives the message, $\mathrm{Sim}_{\mathrm{SC}}$ generates parties Init and Rec in his simulation world to generate the real world situation in Init and Rec exchange messages using $\mathrm{F}_{\mathrm{DIC}}$. $\mathrm{Sim}_{\mathrm{SC}}$ then generates the establish session in the simulation world. That is, he inputs messages $\mathit{in}(\mathtt{Establish}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Init}}$ and $\mathit{in}(\mathtt{Establish}_{\mathtt{SC}}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Rec}}$ to $\mathrm{Init}_{\mathrm{SC}}$ and $\mathrm{Rec}_{\mathrm{SC}}$, respectively. Finally, the parties establish DIC in the simulation world.

Simulation Policy

i. After receiving $\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{F}_{\mathrm{SC}}}$, $\mathrm{Sim}_{\mathrm{SC}}$ executes the following simulation.

   A. $\mathrm{Sim}_{\mathrm{SC}}$ prepares dummy parties, $\mathrm{Init}_{\mathrm{SC}}$, $\mathrm{Rec}_{\mathrm{SC}}$, and Adv and the ideal functionality task $\mathrm{F}_{\mathrm{DIC}}$.

   B. $\mathrm{Sim}_{\mathrm{SC}}$ inputs messages $\mathit{in}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}_{\mathrm{SC}}}$ and $\mathit{in}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Rec}_{\mathrm{SC}}}$ to $\mathrm{Init}_{\mathrm{SC}}$ and $\mathrm{Rec}_{\mathrm{SC}}$, respectively.

   C. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{Init}_{\mathrm{SC}}$ (resp., $\mathrm{Rec}_{\mathrm{SC}}$) send message $\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$.

   D. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{F}_{\mathrm{DIC}}$ send $\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ to Adv.

Task Correspondence of Simulation

i. $\mathrm{Init}_{\mathrm{SC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
   $=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Init}_{\mathrm{SC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

   pre: $\mathit{ntask} = \mathrm{ESS2}$; (M);

   eff: $\mathit{active} := \top$ and $\mathit{ntask} := \bot$; $(L), (M)$;

ii. $\mathrm{Rec}_{\mathrm{SC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
   $=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

   pre: $\mathit{ntask} = \mathrm{ESS2}$; (Q);

   eff: $\mathit{active} = \top$ and $\mathit{ntask} := \bot$; $(P), (Q)$;

iii. $\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Adv}}$
   $=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{SID}, \mathtt{sid}_{\mathtt{SC}})_{\mathrm{Adv}}$

   pre: $\mathit{active} = \top$ and $\mathit{ntask} = \mathrm{ESS2}$; $(G), (I)$;

eff: $ntask := \perp$; (I);

The state *active* of $\mathrm{Init}_{\mathrm{SC}}$ and $\mathrm{Rec}_{\mathrm{SC}}$ becomes $\top$, then state correspondences (L) and (P) hold. If the adversary obtains the message $receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ in the simulation world, *active* becomes $\top$ from (R) in Table 7.27. The simulation in the establish session of the real world is perfectly executed by $\mathrm{Sim}_{\mathrm{SC}}$. Finally, the parties establish a DIC in the simulation world.

(b) **Data Sending Session** Next, in the data sending session, $\mathrm{Sim}_{\mathrm{SC}}$ simulates the key exchange.

Key Exchange

First, the key exchange is executed, that is, the simulator inputs $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}_{\mathrm{DIC}}}$ in Init after a random message is generated using $rand(s)_{sval}$. The receiver is also receives $receive(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}_{\mathrm{DIC}}}$. The receiver generates a random message using $rand(t)_{tval}$ and sends $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F}_{\mathrm{DIC}}}$ to Init. This key exchange is under master schedule $M_{\pi_{\mathrm{SC}}}$. If the master schedule does not be scheduled, then the key exchange does not occurre safely. If so, the adversary can identify the direction in which the random message was sent. That is, he may obtain the key information. Therefore, we need the master schedule. The following describe the corresponding tasks in the key exchange.

Simulation Policy

i. the key exchange is executed as described hereafter.

   A. $\mathrm{Sim}_{\mathrm{SC}}$ executes $random(*)$ and selects key bit s, t, where $|s| = |t| = 1$.

   B. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{Init}_{\mathrm{SC}}$ send $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}_{\mathrm{DIC}}}$ and makes $\mathrm{Rec}_{\mathrm{SC}}$ send $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F}_{\mathrm{DIC}}}$ in random order according to $M_{\pi_{\mathrm{SC}}}$.

   C. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{F}_{\mathrm{DIC}}$ receive $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{Init}_{\mathrm{SC}}}$ and makes $\mathrm{F}_{\mathrm{DIC}}$ send $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{Adv}}$ to Adv.

D. Sim_{SC} makes F_{DIC} receive $\textit{receive}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, t)_{\text{Rec}_{\text{SC}}}$ and makes F_{DIC} send $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, t)_{\text{Adv}}$ to Adv.

E. If F_{DIC} receives $\textit{send}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{F}_{\text{DIC}}}$ from Adv twice, Sim_{SC} continues the following.

F. Sim_{SC} makes F_{DIC} receive $\textit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{Adv}}$ and makes F_{DIC} send $\textit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, t)_{\text{Init}_{\text{SC}}}$ and $\textit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{Rec}_{\text{SC}}}$ to Init_{SC} and Rec_{SC}, respectively.

G. Sim_{SC} makes Init_{SC} and Rec_{SC} receive $\textit{receive}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, t)_{\text{F}_{\text{DIC}}}$ and receive $\textit{receive}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}_{\text{DIC}}}$.

H. Sim_{SC} makes Init_{SC} and Rec_{SC} execute $\textit{keycalc}(sval, tval)_{kval}$, respectively.

I. In $\textit{keycalc}(sval, tval)_{kval}$, if $s \neq t$ then Sim_{SC} continuew next step. Else, Sim_{SC} executes 1(b)iA $\sim$ 1(b)iH until $s \neq t$.

Task Correspondence of Simulation

i. $\text{Init}_{\text{SC}}.\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}_{\text{DIC}}}$
$=_{\text{corr.}}$ $\text{Sim}_{\text{SC}}.\text{Init}_{\text{SC}}.\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}_{\text{DIC}}}$
The precondition and effect of these tasks are identical to each other. Let $\texttt{T}_{\texttt{REAL}}$ and $\texttt{T}_{\texttt{IDEAL}}$ be $\textit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}_{\text{DIC}}}$ and $\textit{keycalc}(sval, tval)_{kval}$. We show that $\texttt{T}_{\texttt{REAL}}$ and $\texttt{T}_{\texttt{IDEAL}}$ are uniformly enabled or disabled in $\text{supp.lst}(\epsilon_{\text{R}}) \cup \text{supp.lst}(\epsilon_{\text{I}})$. We consider that for each state in $s \in \text{supp.lst}(\epsilon_{\text{R}})$ and $u \in \text{supp.lst}(\epsilon_{\text{I}})$ are fixed. $\texttt{T}_{\texttt{REAL}}$ (resp., $\texttt{T}_{\texttt{IDEAL}}$) is enabled (or disabled) if and only if $s.\text{Init}_{\text{SC}}.\textit{ntask} = \text{DSS2}$ (resp., $u.\overline{\text{Init}_{\text{SC}}}.\textit{ntask} = \text{DSS2}$). ($J$) in Table 7.26 implies that $\texttt{T}_{\texttt{REAL}}$ and $\texttt{T}_{\texttt{IDEAL}}$ are uniformly enabled or disabled in $\text{supp.lst}(\epsilon_{\text{R}}) \cup \text{supp.lst}(\epsilon_{\text{I}})$. So, these tasks are activated under the same conditions.

A. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\text{R}} = \epsilon_{\text{R}}$ and $\epsilon'_{\text{I}} = \epsilon_{\text{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (J) in Table 7.26. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

B. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and ideal world, respectively. From each task definitions, the state correspondence of pre: (J), and state correspondence of eff: (J) in Table 7.26, we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

ii. $\mathrm{Rec}_{\mathrm{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}}$
$=_{\mathrm{corr.}} \mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}}$
This is identical to 1(b)i. The state of precondition and effect of both expression are same. . The rest of this proof is similar to 1(b)i.

A. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (N) in Table 7.27. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

B. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definitions, the state correspondence of pre: (N) in Table 7.27, and state correspondence of eff: (N), we have that $\epsilon'_{\mathrm{R}} = \epsilon_{\mathrm{R}}$ and $\epsilon'_{\mathrm{I}} = \epsilon_{\mathrm{I}}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

iii. $\mathrm{Init}_{\mathrm{SC}}.keycalc(sval, tval)_{kval}$
$=_{\mathrm{corr.}} \mathrm{Sim}_{\mathrm{SC}}.\mathrm{Init}_{\mathrm{SC}}.keycalc(sval, tval)_{kval}$

These preconditions and effects are identical to each other. Let $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ be $\mathrm{Init_{SC}}.keycalc(sval, tval)_{kval}$ and $\mathrm{Sim_{SC}}.\mathrm{Init_{SC}}.keycalc(sval, tval)_{kval}$. We show that $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_\mathrm{R}) \cup \mathrm{supp.lst}(\epsilon_\mathrm{I})$. We consider that for each state in $s \in \mathrm{supp.lst}(\epsilon_\mathrm{R})$ and $u \in \mathrm{supp.lst}(\epsilon_\mathrm{I})$ are fixed. $\mathtt{T_{REAL}}$ (resp., $\mathtt{T_{IDEAL}}$) is enabled (or disabled) if and only if $s.\mathrm{Init_{SC}}.ntask = \mathrm{DSSc}$ (resp., $u.\mathrm{Sim_{SC}}.\mathrm{Init_{SC}}.ntask = \mathrm{DSSc}$). (M) in Table 7.26 implies that $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_\mathrm{R}) \cup \mathrm{supp.lst}(\epsilon_\mathrm{I})$. So, these tasks are activated under the same conditions.

A. Disable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (M) in Table 7.26. Therefore, we obtain $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

B. Enable Case: Let ϵ_R and ϵ_I be discrete probability measures in the real world and ideal world, respectively. From each task definition, the state correspondence of pre: (M), and state correspondences of eff: (M) and (U), we have that $\epsilon'_\mathrm{R} = \epsilon_\mathrm{R}$ and $\epsilon'_\mathrm{I} = \epsilon_\mathrm{I}$. Here, the start and step conditions of simulation relation R hold. Therefore, we obtain $trace(\epsilon'_\mathrm{R}) = trace(\epsilon'_\mathrm{I})$.

iv. $\mathrm{Rec_{SC}}.keycalc(sval, tval)_{kval}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim_{SC}}.\mathrm{Rec_{SC}}.keycalc(sval, tval)_{kval}$
The precondition and effect of these tasks are identical to each other. Let $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ be $\mathrm{Init_{SC}}.keycalc(sval, tval$ $)_{kval}$ and $\mathrm{Sim_{SC}}.\mathrm{Init_{SC}}.keycalc(sval, tval)_{kval}$. We show that $\mathtt{T_{REAL}}$ and $\mathtt{T_{IDEAL}}$ are uniformly enabled or disabled in $\mathrm{supp.lst}(\epsilon_\mathrm{R}) \cup \mathrm{supp.lst}(\epsilon_\mathrm{I})$. We consider that for each

state in $s \in \text{supp.lst}(\epsilon_{\text{R}})$ and $u \in \text{supp.lst}(\epsilon_{\text{I}})$ are fixed. $\texttt{T}_{\texttt{REAL}}$ (resp., $\texttt{T}_{\texttt{IDEAL}}$) is enabled (or disabled) if and only if $s.\text{Rec}_{\text{SC}}.\mathit{ntask} = \text{DSSc}$ (resp., $u.\text{Sim}_{\text{SC}}.\text{Rec}_{\text{SC}}.\mathit{ntask} = \text{DSSc}$). ($Q$) in Table 7.27 implies that $\texttt{T}_{\texttt{REAL}}$ and $\texttt{T}_{\texttt{IDEAL}}$ are uniformly enabled or disabled in $\text{supp.lst}(\epsilon_{\text{R}}) \cup \text{supp.lst}(\epsilon_{\text{I}})$. So, these tasks are activated under the same conditions.

A. Disable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. We have the fact that $\epsilon'_{\text{R}} = \epsilon_{\text{R}}$ and $\epsilon'_{\text{I}} = \epsilon_{\text{I}}$. Here, the start and step conditions of simulation relation R hold from each task definition and the state correspondence of pre: (Q) in Table 7.27. Therefore, we obtain $\mathit{trace}(\epsilon'_{\text{R}}) = \mathit{trace}(\epsilon'_{\text{I}})$.

B. Enable Case: Let ϵ_R and ϵ_{I} be discrete probability measures in the real world and the ideal world, respectively. From each task definition, the state correspondence of pre: (Q), and state correspondences of eff: (Q) and (V), we have that $\epsilon'_{\text{R}} = \epsilon_{\text{R}}$ and $\epsilon'_{\text{I}} = \epsilon_{\text{I}}$. Here, the start and step conditions of the simulation relation R hold. Therefore, we obtain $\mathit{trace}(\epsilon'_{\text{R}}) = \mathit{trace}(\epsilon'_{\text{I}})$.

If the key exchange is completed safety, the message exchange proceeds. Init_{SC} sends $\mathit{send}(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, \mathit{cipher})_{\text{F}_{\text{DIC}}}$ to Rec_{SC}. Here, the key exchange is executed under master schedule $M_{\pi_{\text{SC}}}$. If the master schedule does not work, then the key exchange is not safe. If so, the adversary can identify the direction of the random bit although we use F_{DIC}. The details of this task sequence are shown in Table 7.29. This task sequences are also simulated by Sim_{SC}. So, the state correspondences in Tables 7.26, and 7.27 hold. More specifically, Env sends message $\mathit{in}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Init}}}$ (or $\mathit{in}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Rec}}}$) to $\overline{\text{Init}_{\text{SC}}}$ (or $\overline{\text{Rec}_{\text{SC}}}$). $\overline{\text{Init}_{\text{SC}}}$ sends $\mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$ to F_{SC}. F_{SC} sends $\mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, |m|)_{\text{Adv}}$ to Sim_{SC}. After receiving the message, $\mathrm{Sim}_{\mathrm{SC}}$ executes a simulation to mimic the data sending session in the real world. That is, he inputs message $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, m)_{\mathrm{Init}}$ for random message from $\mathrm{F}_{\mathtt{SRC}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, m)_{\mathrm{Rec}}$) to Init (and Rec) in the simulation world. The state correspondences in Table 7.26 and 7.27 work well. The key point of this simulation is as follows. To mimic the real world, the simulator executes the parties that execute the tasks of key exchange (and message sending) in the real world. Moreover, not to distinguish the output trace, the simulator simulates the real world in his simulation world by using task codes. In the real world, $\mathrm{Init}_{\mathrm{SC}}$ and $\mathrm{Rec}_{\mathrm{SC}}$ use a DIC without an adversary being able to identify the direction of the key exchange under master schedule M_{SC}. In the simulation world, $\mathrm{Sim}_{\mathrm{SC}}$ obtains the same output which $\mathrm{Adv}_{\mathrm{SC}}$ outputs in the real world by his simulation. That is, the trace distributions of each world are indistinguishable by Env. In other words, since each task correspondence and state correspondence works well, the following property works well, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$.

Simulation Policy

i. After receiving $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{SC}}, |m|)_{\mathrm{F}_{\mathrm{SC}}}$, $\mathrm{Sim}_{\mathrm{SC}}$ executes the following. simulation.

   A. $\mathrm{Sim}_{\mathrm{SC}}$ generates random bit x ($|x| = 1$) by executing $random(*)$ for using as a message in the simulation.

   B. $\mathrm{Sim}_{\mathrm{SC}}$ inputs $in(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, x)_{\mathrm{Init}_{\mathrm{SC}}}$ to $\mathrm{Init}_{\mathrm{SC}}$.

   C. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{Init}_{\mathrm{SC}}$ send $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, cipher)_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$, where $cipher := x \oplus kval$.

   D. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{F}_{\mathrm{DIC}}$ receive $receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, cipher)_{\mathrm{Init}_{\mathrm{SC}}}$ and makes $\mathrm{F}_{\mathrm{DIC}}$ send $send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, cipher)_{\mathrm{Adv}}$ to Adv.

   E. If $\mathrm{F}_{\mathrm{DIC}}$ receives $send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$ from Adv, $\mathrm{Sim}_{\mathrm{SC}}$ continues the following.

   F. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{F}_{\mathrm{DIC}}$ receive $receive(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok$

$)_{\mathrm{Adv}}$ and makes $\mathrm{F_{DIC}}$ send $\mathit{send}(\mathtt{Receive}, \mathtt{sid_{DIC}}, \mathit{cipher})_{\mathrm{Rec_{SC}}}$ to $\mathrm{Rec_{SC}}$.

G. $\mathrm{Sim_{SC}}$ makes $\mathrm{Rec_{SC}}$ receive the message $\mathit{receive}(\mathtt{Receive}, \mathtt{sid_{DIC}}, \mathit{cipher})_{\mathrm{F_{DIC}}}$.

H. $\mathrm{Sim_{SC}}$ makes $\mathrm{Rec_{SC}}$ output $\mathit{out}(\mathtt{Receive}, \mathtt{sid_{DIC}}, m)_{\mathrm{Rec_{SC}}}$.

ii. $\mathrm{Sim_{SC}}$ executes $\mathit{send}(\mathtt{Response}, \mathtt{sid_{SC}}, \mathit{ok})_{\mathrm{F_{SC}}}$.

Task Correspondence of Simulation

i. $\mathrm{Init_{SC}}.\mathit{send}(\mathtt{Send}, \mathtt{sid_{DIC}}, \mathit{cipher})_{\mathrm{F_{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim_{SC}}.\mathrm{Init_{SC}}.\mathit{send}(\mathtt{Send}, \mathtt{sid_{DIC}}, \mathit{cipher})_{\mathrm{F_{DIC}}}$

pre: $\mathit{active} = \top$, smes and $\mathit{ntask} = \perp$; $(J), (L), (M)$;

eff: $\mathit{smes} := m$ and $\mathit{ntask} :=$ DSS2 ; $(J), (M)$;

ii. $\mathrm{F_{DIC}}.\mathit{send}(\mathtt{Send}, \mathtt{sid_{DIC}}, m)_{\mathrm{Adv_{SC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim_{SC}}.\mathrm{F_{DIC}}.\mathit{send}(\mathtt{Send}, \mathtt{sid_{DIC}}, m)_{\mathrm{Adv_{SC}}}$

pre: $\mathit{okcond}_{\mathrm{Adv}} = \perp$, $\mathit{mes} := m$ and $\mathit{ntask} =$ DSS2 ; $(F), (H), (I)$;

eff: $\mathit{ntask} :=$ DSS3 ; (I);

iii. $\mathrm{Adv_{SC}}.\mathit{send}(\mathtt{Response}, \mathtt{sid_{DIC}}, \mathit{ok})_{\mathrm{F_{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim_{SC}}.\mathrm{Adv_{SC}}.\mathit{send}(\mathtt{Response}, \mathtt{sid_{DIC}}, \mathit{ok})_{\mathrm{F_{DIC}}}$

pre: $\mathit{ntask} =$ DSS2; (T);

eff: $\mathit{ntask} := \perp$; (T);

iv. $\mathrm{F_{DIC}}.\mathit{send}(\mathtt{Receive}, \mathtt{sid_{DIC}}, \mathit{mes})_{\mathrm{Rec_{SC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim_{SC}}.\mathrm{F_{DIC}}.\mathit{send}(\mathtt{Receive}, \mathtt{sid_{DIC}}, \mathit{mes})_{\mathrm{Rec_{SC}}}$

pre: $\mathit{ntask} =$ DSS4 ; (I);

eff: $\mathit{okcond}_{\mathrm{Adv}}$, mes and $\mathit{ntask} := \perp$; $(F), (H), (I)$;

v. $\mathrm{Rec_{SC}}.\mathit{out}(\mathtt{Receive}, \mathtt{sid_{SC}}, \mathit{plain})_{\mathrm{Rec_{SC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim_{SC}}.\mathrm{Rec_{SC}}.\mathit{out}(\mathtt{Receive}, \mathtt{sid_{SC}}, \mathit{plain})_{\mathrm{Rec_{SC}}}$

pre: $\mathit{plain} := \mathit{rmes}$ and $\mathit{ntask} =$ DSS4 ; $(O), (Q)$;

eff: $\mathit{kval}, \mathit{rmes}$ and $\mathit{ntask} := \perp$; $(O), (P), (V)$;

(c) **Expire Session** Finally, in the expire session, Env sends message $\mathit{in}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\overline{\mathrm{Init}}}$ and $\mathit{in}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\overline{\mathrm{Rec}}}$ to $\overline{\mathrm{Init}_{\mathrm{SC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{SC}}}$, respectively. They relay message $\mathit{send}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$ to F_{SC}. After receiving $\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$ from F_{SC}, $\mathrm{Sim}_{\mathrm{SC}}$ expires the session in the simulation world. That is, he inputs messages $\mathit{in}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Init}}$ and $\mathit{in}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Rec}}$ to Init and Rec in the simulation world.

Simulation Policy

i. After receiving $\mathit{receive}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$, $\mathrm{Sim}_{\mathrm{SC}}$ executes the following simulation.

A. $\mathrm{Sim}_{\mathrm{SC}}$ inputs messages, $\mathit{in}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}_{\mathrm{SC}}}$ and $\mathit{in}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Init}_{\mathrm{SC}}}$, to $\mathrm{Init}_{\mathrm{SC}}$ and $\mathrm{Rec}_{\mathrm{SC}}$, respectively.

B. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{Init}_{\mathrm{SC}}$ (resp., $\mathrm{Rec}_{\mathrm{SC}}$) send $\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$.

C. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{F}_{\mathrm{DIC}}$ send $\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$ to Adv.

Task Correspondence of Simulation

i. $\mathrm{Init}_{\mathrm{SC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Init}_{\mathrm{SC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

pre: $\mathit{ntask} = \mathrm{EXS2}$; (L);

eff: active and $\mathit{ntask} := \perp$; $(L), (M)$;

ii. $\mathrm{Rec}_{\mathrm{SC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

pre: $\mathit{ntask} = \mathrm{EXS2}$; (Q);

eff: active and $\mathit{ntask} := \perp$; $(P), (Q)$;

iii. $\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$
$=_{\mathrm{corr.}}$ $\mathrm{Sim}_{\mathrm{SC}}.\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{Adv}}$

pre: $\mathit{ntask} = \mathrm{EXS2}$; (I);

eff: *active*, $estcond_X$ and *ntask* := ⊥ for all X ; $(D) \sim (G)$, (I);

We assume that the state correspondences in Table 7.26 and 7.27 hold. From 3a, 3b, and 3c, the state correspondences also hold after the simulation by Sim_{SC}. That is, $trace(\epsilon'_{\text{R}}) = trace(\epsilon'_{\text{I}})$.

2. **Static Corruption**

   This type of corruption is divided into the following three cases: only Init is corrupted by Adv, only Rec is corrupted by Adv, and both parties are corrupted by Adv. Once the corruption occurs, the adversary can identify the direction. However, the simulator can simulate all the cases, so Env can not distinguish the real world from the ideal world.

   (a) Only Init is corrupted by Adv

   This case means that Adv_{SC} corrupts only Init before the protocol starts. So, the remaining steps are identical to the above-mentioned No Corruption Case without the data sending session. In the data sending session, Adv_{SC} and Sim_{SC} identify the input message.

   After receiving the corrupt message from Env, Sim_{SC} corrupts $\overline{\text{Init}}_{\text{SC}}$ and prepares a simulation world in which only Init_{SC} is corrupted. That is, receiving "corrupt message" from Env, Sim_{SC} corrupts $\overline{\text{Init}}_{\text{SC}}$ and reflects the information in his simulation world immediately. The establish and expire sessions are the same as those in 1a and 1c. The simulation of the data sending session is as follows.

   The simulation policy of the key exchange is same above-mentioned. The simulation policy for the remaining messages sending in the data sending session is as follows:

   Case 1: The message is input to the corrupted party Init_{SC}**.**

   i. After receiving $receive(\texttt{Send}, \texttt{sid}_{\text{SC}}, |m|)_{\text{F}_{\text{SC}}}$, Sim_{SC} executes the following. simulation.

A. Sim_{SC} prepares message m to input to corrupted party $\overline{\text{Init}_{\text{SC}}}$ with $in(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{Init}}$ from Env and reflects m in his simulation world hereafter.

B. Sim_{SC} inputs $in(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}_{\text{SC}}}$ to Init_{SC}.

C. Sim_{SC} makes Init_{SC} send $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$ to F_{DIC}, where $cipher := m \oplus kval$.

D. Sim_{SC} makes F_{DIC} receive $receive(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Init}_{\text{SC}}}$ and makes F_{DIC} send $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Adv}}$ to Adv.

E. If F_{DIC} receives $send(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{F}_{\text{DIC}}}$ from Adv, Sim_{SC} continues the following.

F. Sim_{SC} makes F_{DIC} receive $receive(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{Adv}}$ and makes F_{DIC} send $send(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Rec}_{\text{SC}}}$ to Rec_{SC}.

G. Sim_{SC} makes Rec_{SC} receive $receive(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$.

H. Sim_{SC} makes Rec_{SC} output $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Rec}_{\text{SC}}}$.

ii. After receiving $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Rec}_{\text{SC}}}$ from Rec_{SC}, Sim_{SC} executes $send(\texttt{Response}, \texttt{sid}_{\text{SC}}, ok)_{\text{F}_{\text{SC}}}$.

Case 2: The message is input to the non-corrupted party Rec_{SC}.

i. After receiving $receive(\texttt{Send}, \texttt{sid}_{\text{SC}}, |m|)_{\text{F}_{\text{SC}}}$, Sim_{SC} executes $send(\texttt{Response}, \texttt{sid}_{\text{SC}}, ok)_{\text{F}_{\text{SC}}}$.

ii. After receiving $receive(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$ in $\overline{\text{Init}_{\text{SC}}}$, Sim_{SC} executes the following simulation.

A. Sim_{SC} inputs $in(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}_{\text{SC}}}$ to Rec_{SC}.

B. Sim_{SC} makes Rec_{SC} send $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$ to F_{DIC}, where $cipher := m \oplus kval$.

C. Sim_{SC} makes F_{DIC} receive $receive(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Rec}_{\text{SC}}}$ and makes F_{DIC} send $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Adv}}$ to Adv.

D. If F_{DIC} receives $\mathit{send}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, \mathit{ok})_{F_{DIC}}$ from Adv, Sim_{SC} continues the following.

E. Sim_{SC} makes F_{DIC} receive $\mathit{receive}(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, \mathit{ok})_{\mathrm{Adv}}$ and makes F_{DIC} send $\mathit{send}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, \mathit{cipher})_{\mathrm{Init}_{SC}}$ to Init_{SC}.

F. Sim_{SC} makes Init_{SC} receive $\mathit{receive}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, \mathit{cipher})_{F_{DIC}}$.

G. Sim_{SC} makes Init_{SC} output $\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{Init}_{SC}}$.

iii. After receiving $\mathit{out}(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\mathrm{Init}_{SC}}$ from Init_{SC} in his simulation world, Sim_{SC} executes $\mathit{out}(\texttt{Receive}, \texttt{sid}_{SC}, m)_{\overline{\mathrm{Init}_{SC}}}$.

iv. After receiving $\mathit{receive}(\texttt{Send}, \texttt{sid}_{SC}, m)_{F_{SC}}$ in $\overline{\mathrm{Init}_{SC}}$, Sim_{SC} executes $\mathit{out}(\texttt{Receive}, \texttt{sid}_{SC}, m)_{\overline{\mathrm{Init}_{SC}}}$.

If the message is input to Init or Rec, the simulator emulates the real world and movement of Adv. That is, the simulation is perfectly executed by Sim_{SC}. From the Task Correspondence in 7.3.2, the state correspondences in 7.25, 7.26, and 7.27 hold in this case. That is, $\mathit{trace}(\epsilon'_R) = \mathit{trace}(\epsilon'_I)$ holds.

(b) Only Rec is corrupted by Adv

This case is analogous to the case of a. This case means that Adv_{SC} corrupts only Rec before the protocol starts. So, the remaining steps are identical to the above-mentioned case where only Init is corrupted. Adv_{SC} and Sim_{SC} can identify the input message. So, the simulation is perfectly executed.

After receiving the corrupt message from Env, Sim_{SC} corrupts $\overline{\mathrm{Rec}_{SC}}$ and prepares a simulation world in which only Rec_{SC} is corrupted. That is, receiving "corrupt message" from Env, Sim_{SC} corrupts $\overline{\mathrm{Rec}_{SC}}$ and reflects the information into his simulation world immediately. The establish and expire sessions are the same as those in 1a and 1c. The simulation of the data sending session is as follows.

The simulation policy of the key exchange is the same as that mentioned above. The simulation policy of the left in the data sending session is as follows.

Case 1: The message is input to $\mathrm{Init}_{\mathrm{SC}}$**.**

i. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{F}_{\mathrm{SC}}}$, $\mathrm{Sim}_{\mathrm{SC}}$ executes $\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, ok)_{\mathrm{F}_{\mathrm{SC}}}$.

ii. After receiving $\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$ in the corrupted $\overline{\mathrm{Init}_{\mathrm{SC}}}$, $\mathrm{Sim}_{\mathrm{SC}}$ executes the following simulation.

   A. $\mathrm{Sim}_{\mathrm{SC}}$ inputs $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}_{\mathrm{SC}}}$ to $\mathrm{Init}_{\mathrm{SC}}$.

   B. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{Init}_{\mathrm{SC}}$ send $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$, where $\mathit{cipher} := m \oplus \mathit{kval}$.

   C. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{F}_{\mathrm{DIC}}$ receive $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{Init}_{\mathrm{SC}}}$ and makes $\mathrm{F}_{\mathrm{DIC}}$ send $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{Adv}}$ to Adv.

   D. If $\mathrm{F}_{\mathrm{DIC}}$ receives $\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$ from Adv, $\mathrm{Sim}_{\mathrm{SC}}$ continues the following.

   E. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{F}_{\mathrm{DIC}}$ receive $\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{Adv}}$ and makes $\mathrm{F}_{\mathrm{DIC}}$ send $\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{Init}_{\mathrm{SC}}}$ to $\mathrm{Init}_{\mathrm{SC}}$.

   F. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{Init}_{\mathrm{SC}}$ receive $\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{F}_{\mathrm{DIC}}}$.

   G. $\mathrm{Sim}_{\mathrm{SC}}$ makes $\mathrm{Init}_{\mathrm{SC}}$ output $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Rec}_{\mathrm{SC}}}$.

iii. After receiving $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}_{\mathrm{SC}}}$ from $\mathrm{Init}_{\mathrm{SC}}$ in his simulation world, executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\overline{\mathrm{Init}_{\mathrm{SC}}}}$.

iv. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$ in $\overline{\mathrm{Rec}_{\mathrm{SC}}}$, $\mathrm{Sim}_{\mathrm{SC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\overline{\mathrm{Rec}_{\mathrm{SC}}}}$.

Case 2: The message is input to corrupted $\mathrm{Rec}_{\mathrm{SC}}$**.**

i. After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{F}_{\mathrm{SC}}}$, $\mathrm{Sim}_{\mathrm{SC}}$ executes the following. simulation.

A. Sim_{SC} prepares message m to input to corrupted party $\overline{\text{Rec}_{\text{SC}}}$ with $in(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{Rec}}$ from Env and reflects m in his simulation world hereafter.

B. Sim_{SC} inputs $in(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}_{\text{SC}}}$ to Init_{SC}.

C. Sim_{SC} makes Init_{SC} send $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$ to F_{DIC}, where $cipher := m \oplus kval$.

D. Sim_{SC} makes F_{DIC} receive $receive(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Init}_{\text{SC}}}$ and makes F_{DIC} send $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Adv}}$ to Adv.

E. If F_{DIC} receives $send(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{F}_{\text{DIC}}}$ from Adv, Sim_{SC} continues the following.

F. Sim_{SC} makes F_{DIC} receive $receive(\texttt{Response}, \texttt{sid}_{\texttt{DIC}}, ok)_{\text{Adv}}$ and makes F_{DIC} send $send(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{Init}_{\text{SC}}}$ to Init_{SC}.

G. Sim_{SC} makes Init_{SC} receive $receive(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$.

H. Sim_{SC} makes Init_{SC} output $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}_{\text{SC}}}$.

ii. After receiving $out(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, m)_{\text{Init}_{\text{SC}}}$ from Init_{SC}, Sim_{SC} executes $send(\texttt{Response}, \texttt{sid}_{\text{SC}}, ok)_{\text{F}_{\text{SC}}}$.

If the message is input to Init or Rec, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by Sim_{SC}. From the Task Correspondence in 7.3.2, the state correspondences in 7.25, 7.26, and 7.27 hold in this case. That is, $trace(\epsilon'_{\text{R}}) = trace(\epsilon'_{\text{I}})$ holds.

(c) Both parties are corrupted by Adv

This case is also analogous to case 1 of a. This case means that Adv_{SC} corrupts both Init and Rec before the protocol starts. After receiving the corrupt message from Env, Sim_{SC} corrupts $\overline{\text{Init}_{\text{SC}}}$ and $\overline{\text{Rec}_{\text{SC}}}$ and prepares a simulation world in which $\overline{\text{Init}_{\text{SC}}}$ and $\overline{\text{Rec}_{\text{SC}}}$ are corrupted. That is, after receiving the "corrupt message" from Env, Sim_{SC} corrupts $\overline{\text{Init}_{\text{SC}}}$ and $\overline{\text{Rec}_{\text{SC}}}$ and reflects the information in his simulation world, immediately. The establish and expire sessions are same as 1a and 1c. The simulation of the data sending session is as follows:

The simulation policy of the key exchange is the same as that mentioned above. The simulation policy for the remaining messages in the data sending session is as follows:

Case 1: The message is input to corrupted Init_{SC}**.**

This case is identical to Case 1 of a.

Case 2: The message is input to corrupted Rec_{SC}**.**

This case is identical to Case 2 of b.

Sim_{SC} executes the following in the above mentioned cases:

- After receiving $\mathit{receive}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$ in $\mathit{party} \in \{\overline{\text{Init}_{\text{SC}}}, \overline{\text{Rec}_{\text{SC}}}\}$, Sim_{SC} executes $\mathit{out}(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\mathit{party}}$.

If the message is input to Init or Rec, the simulator emulates the real world and the movement of Adv. That is, the simulation is perfectly executed by Sim_{SC}. From the Task Correspondence in 7.3.2, the state correspondences in 7.25, 7.26, and 7.27 hold in this case. That is, $\mathit{trace}(\epsilon'_{\text{R}}) = \mathit{trace}(\epsilon'_{\text{I}})$ holds.

3. **Adaptive Corruption** In this case, the adversary corrupts some parties when he wants to do so. This case is also simulated by the simulator, but the message cannot be concealed from the adversary after he corrupts some parties. However, this case is also simulated by simulator Sim_{SC}, so the simulation is perfectly executed.

   (a) **Establish Session**

   Instance 1: Before Init_{SC} and Rec_{SC} are activated.

   Instance 2: After Init_{SC} is activated but before Rec_{SC} is activated.

   Instance 3: After Rec_{SC} is activated but before Init_{SC} is activated.

   Instance 4: After Init_{SC} and Rec_{SC} are activated.

Because there is no secret information, $\mathrm{Sim}_{\mathrm{SC}}$ can emulate the situations in his simulation. So, there is no advantage for Env. The adversary can corrupt $\mathrm{Init}_{\mathrm{SC}}$, $\mathrm{Rec}_{\mathrm{SC}}$, or both, but the simulator can also corrupt the corresponding dummy parties. These cases are also perfectly simulated by $\mathrm{Sim}_{\mathrm{SC}}$.

(b) **Data Sending Session**

Instance 1: Before or after activating $\mathrm{Init}_{\mathrm{SC}}$ or $\mathrm{Rec}_{\mathrm{SC}}$ by receiving $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Init}_{\mathrm{SC}}}$ or $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Rec}_{\mathrm{SC}}}$, respectively, from Env.
Env can execute only the message sending indication or the corrupt indication at a time. This case is also simulated by $\mathrm{Sim}_{\mathrm{SC}}$ as the data sending session of Static Corruption without the corruption timing. If the corruption message is received from Env, Adv and Sim corrupt the party, and then they continue the protocol. Note that all information without *active* are cleared after the message is sent. So, there is no secret information. The corresponding tasks work well and there exists a simulation relation between the real world and the ideal world. That is, $trace(\epsilon'_{\mathrm{R}}) = trace(\epsilon'_{\mathrm{I}})$ holds.

(c) **Expire Session**

Instance 1: After $\mathrm{Init}_{\mathrm{SC}}$ or $\mathrm{Rec}_{\mathrm{SC}}$ is activated with the expire message.
Once the expire message is sent to $\mathrm{Init}_{\mathrm{SC}}$ or $\mathrm{Rec}_{\mathrm{SC}}$ by Env, this session terminates in the real world and the ideal world. So the adversary can corrupt the parties. The simulation is also executed.

Simulation Policy

$\mathrm{Sim}_{\mathrm{SC}}$ simulates as follows:

(a) After receiving the “corrupt $\mathrm{Init}_{\mathrm{SC}}$” message from Env,

   i. $\mathrm{Sim_{SC}}$ corrupts $\overline{\mathrm{Init_{SC}}}$ and makes Adv corrupt $\mathrm{Init_{SC}}$ in the simulation world, immediately.

(b) After receiving the "corrupt $\mathrm{Rec_{SC}}$" message from Env,

   i. $\mathrm{Sim_{SC}}$ corrupts $\overline{\mathrm{Rec_{SC}}}$ and makes Adv corrupt $\mathrm{Rec_{SC}}$ in the simulation world, immediately.

(c) After receiving the "corrupt $\mathrm{Init_{SC}}$ and $\mathrm{Rec_{SC}}$" message from Env,

   i. $\mathrm{Sim_{SC}}$ corrupts $\overline{\mathrm{Init_{SC}}}$ and $\overline{\mathrm{Rec_{SC}}}$, and makes Adv corrupt $\mathrm{Init_{SC}}$ and $\mathrm{Rec_{SC}}$ in the simulation world, immediately.

(d) After receiving $\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}}$ from $\mathrm{F_{SC}}$.

   i. $\mathrm{Sim_{SC}}$ inputs message $\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathit{party}}$ to message input party *party* in his simulation.

   ii. The remaining steps are the same as the simulation of the No Corrupted Case.

(e) After receiving $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}}$ in $\overline{\mathit{party}}$, $\mathrm{Sim_{SC}}$ executes $\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\overline{\mathit{party}}}$.

Whenever $\mathrm{Adv_{SC}}$ corrupts some party, $\mathrm{Sim_{SC}}$ corrupts the corresponding dummy party in the ideal world and forwards the obtained information to the simulated copy of $\mathrm{Adv_{SC}}$. If $\mathrm{Adv_{SC}}$ corrupts a party $\mathrm{Init_{SC}}$ or $\mathrm{Rec_{SC}}$ then $\mathrm{Sim_{SC}}$ corrupts $\overline{\mathrm{Init_{SC}}}$ or (and) $\overline{\mathrm{Rec_{SC}}}$ in the ideal world, and provides the simulated copy of $\mathrm{Adv_{SC}}$ in the simulation world with the state information of the corrupted party. Conversely, $\mathrm{Sim_{SC}}$ may obtain information from the simulated world with the corruption. Additionally, in this protocol, the party has no secret information because $\mathrm{F_{DIC}}$ is securely executed. In all cases, since $\mathrm{Sim_{SC}}$ can simulate $\mathrm{Adv_{SC}}$ using his simulated world, Env cannot distinguish real world from the ideal world. That is, simulating party corruption is perfectly executed.

Relation R is a simulation relation from task and state correspondence. We obtain Lemma4.

□

Next, Theorem7 is obtained from Lemma4 immediately.

Proof. From Lemma4, we proved that relation R is a simulation relation from $\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}$ to $\mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}$.

Theorem7 is also proven from Theorem3, that is, we obtain that ϵ_{R} and ϵ_{I} satisfy the trace distribution property, $tdist(\epsilon_{\mathrm{R}}) = tdist(\epsilon_{\mathrm{I}})$.

As a result, the simulation is perfectly executed because simulator $\mathrm{Sim}_{\mathrm{SC}}$ can simulate the real world from the information message through $\mathrm{Adv}_{\mathrm{SC}}$. The tasks of the real world perfectly correspond to the tasks of the ideal world. That is,

$$\mathrm{Real}_{\mathrm{SC}}||\mathrm{Env}\ \ \mathrm{Hyb.} \leq_0^{\mathtt{M}_{\pi\mathrm{SC}}} \mathrm{Ideal}_{\mathrm{SC}}||\mathrm{Env}.$$

□

Functionality

(a)	$u.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Init}} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Init}}$
(b)	$u.\mathrm{F}_{\mathrm{SC}}.\mathit{estcond}_{\mathrm{Rec}} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{estcond}_{\mathrm{Rec}}$
(c)	$u.\mathrm{F}_{\mathrm{SC}}.\mathit{okcond}_{\mathrm{Adv}} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{okcond}_{\mathrm{Adv}}$
(d)	$u.\mathrm{F}_{\mathrm{SC}}.\mathit{active} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{active}$
(e)	$u.\mathrm{F}_{\mathrm{SC}}.\mathit{mes} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{mes}$
(f)	$u.\mathrm{F}_{\mathrm{SC}}.\mathit{ntask} = s.\mathrm{F}_{\mathrm{DIC}}.\mathit{ntask}$

Initiator

(g)	$u.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{smes} = s.\mathrm{Init}_{\mathrm{SC}}.\mathit{smes}$
(h)	$u.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{rmes} = s.\mathrm{Init}_{\mathrm{SC}}.\mathit{rmes}$
(i)	$u.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{active} = s.\mathrm{Init}_{\mathrm{SC}}.\mathit{active}$
(j)	$u.\overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{ntask} = s.\mathrm{Init}_{\mathrm{SC}}.\mathit{ntask}$

Receiver

(k)	$u.\overline{\mathrm{Rec}_{\mathrm{SC}}}.\mathit{smes} = s.\mathrm{Rec}_{\mathrm{SC}}.\mathit{smes}$
(l)	$u.\overline{\mathrm{Rec}_{\mathrm{SC}}}.\mathit{rmes} = s.\mathrm{Rec}_{\mathrm{SC}}.\mathit{rmes}$
(m)	$u.\overline{\mathrm{Rec}_{\mathrm{SC}}}.\mathit{active} = s.\mathrm{Rec}_{\mathrm{SC}}.\mathit{active}$
(n)	$u.\overline{\mathrm{Rec}_{\mathrm{SC}}}.\mathit{ntask} = s.\mathrm{Rec}_{\mathrm{SC}}.\mathit{ntask}$

Environment

(o)	$u.\mathrm{Env} = s.\mathrm{Env}$

Table 7.25: State Correspondence for $\mathrm{Real}_{\mathrm{SC}}$ and $\mathrm{Ideal}_{\mathrm{SC}}$ (Part I)

Simulator (or Adversary)

(A)	$u.\text{Sim}_{\text{SC}}.\mathit{active} = s.\text{Adv}_{\text{SC}}.\mathit{active}$
(B)	$u.\text{Sim}_{\text{SC}}.\mathit{smes} = s.\text{Adv}_{\text{SC}}.\mathit{smes}$
(C)	$u.\text{Sim}_{\text{SC}}.\mathit{ntask} = s.\text{Adv}_{\text{SC}}.\mathit{ntask}$
(D)	$u.\text{Sim}_{\text{SC}}.\text{F}_{\text{DIC}}.\mathit{estcond}_{\text{Init}} = s.\text{F}_{\text{DIC}}.\mathit{estcond}_{\text{Init}}$
(E)	$u.\text{Sim}_{\text{SC}}.\text{F}_{\text{DIC}}.\mathit{estcond}_{\text{Rec}} = s.\text{F}_{\text{DIC}}.\mathit{estcond}_{\text{Rec}}$
(F)	$u.\text{Sim}_{\text{SC}}.\text{F}_{\text{DIC}}.\mathit{okcond}_{\mathit{adv}} = s.\text{F}_{\text{DIC}}.\mathit{okcond}_{\mathit{adv}}$
(G)	$u.\text{Sim}_{\text{SC}}.\text{F}_{\text{DIC}}.\mathit{active} = s.\text{F}_{\text{DIC}}.\mathit{active}$
(H)	$u.\text{Sim}_{\text{SC}}.\text{F}_{\text{DIC}}.\mathit{mes} = s.\text{F}_{\text{DIC}}.\mathit{mes}$
(I)	$u.\text{Sim}_{\text{SC}}.\text{F}_{\text{DIC}}.\mathit{ntask} = s.\text{F}_{\text{DIC}}.\mathit{ntask}$
(J)	$u.\text{Sim}_{\text{SC}}.\text{Init}_{\text{SC}}.\mathit{smes} = s.\text{Init}_{\text{SC}}.\mathit{smes}$
(K)	$u.\text{Sim}_{\text{SC}}.\text{Init}_{\text{SC}}.\mathit{rmes} = s.\text{Init}_{\text{SC}}.\mathit{rmes}$
(L)	$u.\text{Sim}_{\text{SC}}.\text{Init}_{\text{SC}}.\mathit{active} = s.\text{Init}_{\text{SC}}.\mathit{active}$
(M)	$u.\text{Sim}_{\text{SC}}.\text{Init}_{\text{SC}}.\mathit{ntask} = s.\text{Init}_{\text{SC}}.\mathit{ntask}$

Table 7.26: State Correspondence for Real_{SC} and Ideal_{SC} (Part II)

Simulator (or Adversary)

(N)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.smes = s.\mathrm{Rec}_{\mathrm{SC}}.smes$
(O)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.rmes = s.\mathrm{Rec}_{\mathrm{SC}}.rmes$
(P)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.active = s.\mathrm{Rec}_{\mathrm{SC}}.active$
(Q)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.ntask = s.\mathrm{Rec}_{\mathrm{SC}}.ntask$
(R)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Adv}_{\mathrm{SC}}.active = s.\mathrm{Adv}_{\mathrm{SC}}.active$
(S)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Adv}_{\mathrm{SC}}.smes = s.\mathrm{Adv}_{\mathrm{SC}}.smes$
(T)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Adv}_{\mathrm{SC}}.ntask = s.\mathrm{Adv}_{\mathrm{SC}}.ntask$
(U)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Init}_{\mathrm{SC}}.kval = s.\mathrm{Init}_{\mathrm{SC}}.kval$
(V)	$u.\mathrm{Sim}_{\mathrm{SC}}.\mathrm{Rec}_{\mathrm{SC}}.kval = s.\mathrm{Rec}_{\mathrm{SC}}.kval$

Table 7.27: State Correspondence for $\mathrm{Real}_{\mathrm{SC}}$ and $\mathrm{Ideal}_{\mathrm{SC}}$ (Part III)

1. **Establish Session**

(a)	$\mathrm{Init}_{\mathrm{SC}}.\mathit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
	$=_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{send}(\texttt{Establish}_{\mathrm{SC}}, \texttt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
(b)	$\mathrm{Rec}_{\mathrm{SC}}.\mathit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
	$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{SC}}}.\mathit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
(c)	$\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\texttt{SID}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F}_{\mathrm{SC}}.\mathit{send}(\texttt{SID}, \texttt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$

2. **Expire Session**

(a)	$\mathrm{Init}_{\mathrm{SC}}.\mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
	$=_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{SC}}}.\mathit{send}(\texttt{Expire}_{\mathrm{SC}}, \texttt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
(b)	$\mathrm{Rec}_{\mathrm{SC}}.\mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
	$=_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{SC}}}.\mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\mathrm{SC}})_{\mathrm{F}_{\mathrm{SC}}}$
(c)	$\mathrm{F}_{\mathrm{DIC}}.\mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{Adv}}$
	$=_{\mathrm{corr.}} \mathrm{F}_{\mathrm{SC}}.\mathit{send}(\texttt{Expire}_{\mathrm{SC}}, \texttt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$

3. **Environment**

(a)	All tasks of environment Env in $\mathrm{Real}_{\mathrm{SC}}$ are correspond with the tasks of environment in $\mathrm{Ideal}_{\mathrm{SC}}$.

Table 7.28: Corresponding Tasks for $\mathrm{Real}_{\mathrm{SC}}$ and $\mathrm{Ideal}_{\mathrm{SC}}$

Real Execution

No.	$\mathrm{Init}_{\mathrm{SC}}$	$\mathrm{Rec}_{\mathrm{SC}}$	$\mathrm{F}_{\mathrm{DIC}}$	$\mathrm{Adv}_{\mathrm{SC}}$
1	$\mathit{rand}(s)^{\mathrm{Init}}_{\mathit{sval}}$	$\mathit{rand}(t)^{\mathrm{Rec}}_{\mathit{tval}}$		
2	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}_{\mathrm{DIC}}}$	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F}_{\mathrm{DIC}}}$		
3			$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_X$	
4			$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}_{\mathrm{SC}}}$	
5				$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)$
6				$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{DIC}}}$
7			$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{Adv}_{\mathrm{SC}}}$	
8			$\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{bX}$	
9	$\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F}_{\mathrm{DIC}}}$	$\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}_{\mathrm{DIC}}}$		
10	$\mathit{keycalc}(\mathit{sval}, \mathit{tval})_{\mathit{kval}}$	$\mathit{keycalc}(\mathit{sval}, \mathit{tval})_{\mathit{kval}}$		
11	$\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Init}_{\mathrm{SC}}}$			
12	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{F}_{\mathrm{DIC}}}$			
13			$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Init}_{\mathrm{SC}}}$	
14			$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{Adv}_{\mathrm{SC}}}$	
15				$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)$
16				$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{DIC}}}$
17			$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{Adv}_{\mathrm{SC}}}$	
18			$\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{mes})_{\mathrm{Rec}_{\mathrm{SC}}}$	
19		$\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{F}_{\mathrm{DIC}}}$		
20		$\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{plain})_{\mathrm{Rec}_{\mathrm{SC}}}$		

Ideal Execution

No.	$\overline{\mathrm{Init}_{\mathrm{SC}}}$	$\overline{\mathrm{Rec}_{\mathrm{SC}}}$	F_{SC}	$\mathrm{Sim}_{\mathrm{SC}}$
1	$\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\overline{\mathrm{Init}_{\mathrm{SC}}}}$			
2	$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$			
3			$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\overline{\mathrm{Init}_{\mathrm{SC}}}}$	
4			$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, \lvert m\rvert)_{\mathrm{Sim}_{\mathrm{SC}}}$	
5				$\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, \lvert m\rvert)_{\mathrm{F}_{\mathrm{SC}}}$
6				$\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{SC}}}$
7			$\mathit{receive}(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, \mathit{ok})_{\mathrm{Sim}_{\mathrm{SC}}}$	
8			$\mathit{send}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{mes})_{\overline{\mathrm{Rec}_{\mathrm{SC}}}}$	
9		$\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F}_{\mathrm{SC}}}$		
10		$\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, m)_{\overline{\mathrm{Rec}_{\mathrm{SC}}}}$		

Table 7.29: Corresponding Task Sequence of Data Sending Session for $\mathrm{Real}_{\mathrm{SC}}$ and $\mathrm{Ideal}_{\mathrm{SC}}$ under $M_{\pi_{\mathrm{SC}}}$

Code for Initiator of Secure Channel, Init_{SC}

Signature:

$\texttt{sid}_{\texttt{SC}} = (\text{Init}, \text{Rec}, \texttt{sid}'_{\texttt{SC}})$
$\texttt{sid}_{\texttt{DIC}} = (\{\text{Init}, \text{Rec}\}, \texttt{sid}'_{\texttt{DIC}})$

Input:
$in(\texttt{Establish}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{Init}}$
$in(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, m)_{\text{Init}}$
$rand(s)^{\text{Init}}_{sval}$
$receive(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, t)_{\text{F}_{\text{DIC}}}$
$receive(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$
$in(\texttt{Expire}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{Init}}$

Output:
$send(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
$send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}_{\text{DIC}}}$
$send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$
$out(\texttt{Receive}, \texttt{sid}_{\texttt{SC}}, plain)_{\text{Init}}$
$send(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$

Internal:
$keycalc(sval, tval)_{kval}$

State:

$smes, rmes \in \{0,1\} \cup \{\perp\}$, initially $\perp$ $\quad$ $kval \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$
$active \in \{\perp, \top\}$, initially $\perp$ $\quad$ $sval \in \{0,1\} \cup \{\perp\}$, initially $\perp$
$tval \in \{0,1\} \cup \{\perp\}$, initially $\perp$ $\quad$ $ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$

Tasks:

$\{send(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}, send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}_{\text{DIC}}},$
$send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}, out(\texttt{Receive}, \texttt{sid}_{\texttt{SC}}, plain)_{\text{Init}},$
$send(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}, keycalc(sval, tval)_{kval}\}$

Figure 7.45: Code for Initiator of Secure Channel, Init_{SC} (Part I)

Code for Initiator of Secure Channel, Init_{SC}

```
Transitions:

    Establish Session:

        ESS1. $in(\texttt{Establish}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{Init}}$
              pre: $active, ntask = \perp$
              eff: $ntask := \text{ESS2}$
        ESS2. $send(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
              pre: $ntask = \text{ESS2}$
              eff: $active := \top$ and $ntask := \perp$

    Data Sending Session:

        DSS1. $in(\texttt{Send}, \texttt{sid}_{\texttt{SC}}, m)_{\text{Init}}$
              pre: $active = \top$, $smes$ and $ntask = \perp$
              eff: $smes := m$ and $ntask := \text{DSS2}$
        DSS2. $rand(s)^{\text{Init}}_{sval}$
              pre: $active = \top$, $sval, kval$ and $ntask = \perp$
              eff: $sval := s$ and $ntask := \text{DSSa}$
        DSS3. $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, s)_{\text{F}_{\text{DIC}}}$
              pre: $s := sval$ and $ntask = \text{DSSa}$
              eff: $ntask := \text{DSSb}$
        DSS4. $receive(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, t)_{\text{F}_{\text{DIC}}}$
              pre: $tval = \perp$ and $ntask = \text{DSSb}$
              eff: $tval := t$ and $ntask := \text{DSSc}$
```

Figure 7.46: Code for Initiator of Secure Channel, Init_{SC} (Part II)

```
Code for Initiator of Secure Channel, $\text{Init}_{\text{SC}}$

Transitions:

    Data Sending Session:

        DSS5. $keycalc(sval, tval)_{kval}$
            pre: $ntask = \text{DSSc}$
            eff:
            If $sval \neq tval$ then $kval := sval$, and $sval$, $tval$ and $ntask := \perp$.
            Else all values without $active$ set initial value $\perp$.
        DSS6. $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$
            pre: $kval \neq \perp$, $cipher := smes \oplus kval$ and $ntask = \text{DSS2}$
            eff: $smes, cipher$ and $ntask := \perp$
        DSS7. $receive(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\text{F}_{\text{DIC}}}$
            pre: $active$, $kval \neq \perp$, $rmes$ and $ntask = \perp$
            eff: $rmes := cipher \oplus kval$ and $ntask := \text{DSS4}$
        DSS8. $out(\texttt{Receive}, \texttt{sid}_{\texttt{SC}}, plain)_{\text{Init}}$
            pre: $plain := rmes$ and $ntask = \text{DSS4}$
            eff: $kval, rmes$ and $ntask := \perp$

    Expire Session:

        EXS1. $in(\texttt{Expire}_{\texttt{SC}}, \texttt{sid}_{\texttt{SC}})_{\text{Init}}$
            pre: $active = \top$, $mes$ and $ntask = \perp$
            eff: $ntask := \text{EXS2}$
        EXS2. $send(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$
            pre: $ntask = \text{EXS2}$
            eff: $active$ and $ntask := \perp$
```

Figure 7.47: Code for Initiator of Secure Channel, Init_{SC} (Part III)

Code for Receiver of Secure Channel, $\mathrm{Rec}_{\mathrm{SC}}$

Signature:

$\mathtt{sid}_{\mathrm{SC}} = (\mathrm{Init}, \mathrm{Rec}, \mathtt{sid}'_{\mathrm{SC}})$
$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathtt{DIC}})$

Input:
$\mathit{in}(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Rec}}$
$\mathit{in}(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{Rec}}$
$\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{receive}(\mathtt{Receive}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{rand}(t)^{\mathrm{Rec}}_{\mathit{tval}}$
$\mathit{in}(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Rec}}$

Output:
$\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{F}_{\mathrm{DIC}}}$
$\mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{plain})_{\mathrm{Rec}}$
$\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

Internal:
$\mathit{keycalc}(\mathit{sval}, \mathit{tval})_{\mathit{kval}}$

State:

$\mathit{smes}, \mathit{rmes} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$	$\mathit{kval} \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$
$\mathit{active} \in \{\perp, \top\}$, initially $\perp$	$\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$\mathit{sval} \in \{0,1\} \cup \{\perp\}$, initially $\perp$	$\mathit{tval} \in \{0,1\} \cup \{\perp\}$, initially $\perp$

Tasks:

$\{\mathit{send}(\mathtt{Establish}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}, \mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F}_{\mathrm{DIC}}},$
$\mathit{send}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{cipher})_{\mathrm{F}_{\mathrm{DIC}}}, \mathit{out}(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, \mathit{plain})_{\mathrm{Rec}},$
$\mathit{send}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}, \mathit{keycalc}(\mathit{sval}, \mathit{tval})_{\mathit{kval}}\}$

Figure 7.48: Code for Receiver of Secure Channel, $\mathrm{Rec}_{\mathrm{SC}}$ (Part I)

```
Code for Receiver of Secure Channel, $\text{Rec}_{\text{SC}}$

Transitions:

    Establish Session:

        ESS1. $in(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{Rec}}$
                pre: $active, ntask = \bot$
                eff: $ntask := \text{ESS2}$
        ESS2. $send(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$
                pre: $ntask = \text{ESS2}$
                eff: $active := \top$ and $ntask := \bot$

    Data Sending Session:

        DSS1. $in(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{Rec}}$
                pre: $active = \top$, $smes$ and $ntask = \bot$
                eff: $smes := m$ and $ntask := \text{DSS2}$
        DSS2. $rand(t)^{\text{Rec}}_{tval}$
                pre: $active = \top$, $tval, kval$ and $ntask = \bot$
                eff: $tval := t$ and $ntask := \text{DSSa}$
        DSS3. $send(\texttt{Send}, \texttt{sid}_{\text{DIC}}, t)_{\text{F}_{\text{DIC}}}$
                pre: $t := tval$ and $ntask = \text{DSSa}$
                eff: $ntask := \text{DSSb}$
        DSS4. $receive(\texttt{Receive}, \texttt{sid}_{\text{DIC}}, s)_{\text{F}_{\text{DIC}}}$
                pre: $sval = \bot$ and $ntask = \text{DSSb}$
                eff: $sval := s$ and $ntask := \text{DSSc}$

    Expire Session:

        EXS1. $in(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{Rec}}$
                pre: $active = \top$, $smes, rmes$ and $ntask = \bot$
                eff: $ntask := \text{EXS2}$
        EXS2. $send(\texttt{Expire}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$
                pre:$ntask = \text{EXS2}$
                eff: $active$ and $ntask := \bot$
```

Figure 7.49: Code for Receiver of Secure Channel, Rec_{SC} (Part II)

Code for Receiver of Secure Channel, $\mathrm{Rec}_{\mathrm{SC}}$

Transitions:

Data Sending Session:

```
DSS5. $keycalc(sval, tval)_{kval}$
        pre: $ntask = \mathrm{DSSc}$
        eff: If $sval \neq tval$ then $kval := sval$ , and $sval$, $tval$ and $ntask :=$
      $\perp$
        Else all values without $active$ set initial value $\perp$.
DSS6. $send(\texttt{Send}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\mathrm{F_{DIC}}}$
        pre: $kval \neq \perp$, $cipher := smes \oplus kval$ and $ntask = \mathrm{DSS2}$
        eff: $smes, cipher := \perp$ and $ntask := \perp$
DSS7. $receive(\texttt{Receive}, \texttt{sid}_{\texttt{DIC}}, cipher)_{\mathrm{F_{DIC}}}$
        pre: $active$, $kval \neq \perp$, $rmes$ and $ntask = \perp$
        eff: $rmes := cipher \oplus kval$ and $ntask := \mathrm{DSS4}$
DSS8. $out(\texttt{Receive}, \texttt{sid}_{\mathrm{SC}}, plain)_{\mathrm{Rec}}$
        pre: $plain := rmes$ and $ntask = \mathrm{DSS4}$
        eff: $kval, rmes$ and $ntask := \perp$
```

Expire Session:

```
EXS1. $in(\texttt{Expire}_{\mathrm{SC}}, \texttt{sid}_{\mathrm{SC}})_{\mathrm{Rec}}$
        pre: $active = \top$, $smes, rmes$ and $ntask = \perp$
        eff: $ntask := \mathrm{EXS2}$
EXS2. $send(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\mathrm{F_{DIC}}}$
        pre:$ntask = \mathrm{EXS2}$
        eff: $active$ and $ntask := \perp$
```

Figure 7.50: Code for Receiver of Secure Channel, $\mathrm{Rec}_{\mathrm{SC}}$ (Part III)

Code for Adversary for Secure Channel, $\mathrm{Adv}_{\mathrm{SC}}$

Signature:

$\mathtt{sid}_{\mathtt{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathtt{DIC}})$

Input:
$receive(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
$receive(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$
$receive(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$

Output:
$send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$

Other:
*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message and $out(*)$.

State:

$active \in \{\perp, \top\}$, initially $\perp$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$smes \in (\{0,1\}) \cup \{\perp\}$, initially $\perp$

Tasks:

$\{send(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, ok)_{\mathrm{F}_{\mathrm{DIC}}}$, other arbitrary tasks$\}$

Figure 7.51: Code for Adversary for Secure Channel, $\mathrm{Adv}_{\mathrm{SC}}$ (Part I)

Code for Adversary for Secure Channel, $\mathrm{Adv}_{\mathrm{SC}}$

Transitions:

Establish Session:

```
ESS1. $\mathit{receive}(\mathtt{SID}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $\mathit{active} = \bot$
        eff: $\mathit{active} := \top$
```

Data Sending Session:

```
DSS1. $\mathit{receive}(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, m)$
        pre: $\mathit{active} = \top$ and $\mathit{ntask} = \bot$
        eff: $\mathit{smes} := m$ and $\mathit{ntask} :=$ DSS2
DSS2. $\mathit{send}(\mathtt{Response}, \mathtt{sid}_{\mathtt{DIC}}, \mathit{ok})_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $\mathit{ntask} =$ DSS2
        eff: $\mathit{smes}, \mathit{ntask} := \bot$
```

Expire Session:

```
EXS1. $\mathit{receive}(\mathtt{Expire}_{\mathtt{DIC}}, \mathtt{sid}_{\mathtt{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$
        pre: $\mathit{active} = \top$
        eff: $\mathit{active} := \bot$
```

Other tasks:

This adversary makes other arbitary tasks.

Figure 7.52: Code for Adversary for Secure Channel, $\mathrm{Adv}_{\mathrm{SC}}$ (Part II)

Code for ideal Initiator of Secure Channel, $\overline{\text{Init}}_{\text{SC}}$

Signature:

$\mathtt{sid}_{\text{SC}} = (\text{Init}, \text{Rec}, \mathtt{sid}'_{\text{SC}})$

Input:	Output:
$in(\mathtt{Establish}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\overline{\text{Init}}}$	$send(\mathtt{Establish}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
$in(\mathtt{Send}, \mathtt{sid}_{\text{SC}}, m)_{\overline{\text{Init}}}$	$send(\mathtt{Send}, \mathtt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
$receive(\mathtt{Receive}, \mathtt{sid}_{\text{SC}}, mes)_{\text{F}_{\text{SC}}}$	$out(\mathtt{Receive}, \mathtt{sid}_{\text{SC}}, mes)_{\overline{\text{Init}}}$
$in(\mathtt{Expire}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\overline{\text{Init}}}$	$send(\mathtt{Expire}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$

State:

$smes, rmes \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$active \in \{\perp, \top\}$, initially $\perp$

Tasks:

$\{send(\mathtt{Establish}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}, send(\mathtt{Send}, \mathtt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}},$
$send(\mathtt{Expire}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}, out(\mathtt{Receive}, \mathtt{sid}_{\text{SC}}, mes)_{\overline{\text{Init}}}\}$

Figure 7.53: Code for ideal Initiator of Secure Channel, $\overline{\text{Init}}_{\text{SC}}$ (Part I)

Code for ideal Initiator of Secure Channel, $\overline{\text{Init}}_{\text{SC}}$

```
Transitions:

    Establish Session:

        ESS1. $in(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\overline{\text{Init}}}$
              pre: $active, ntask = \bot$
              eff: $ntask :=$ ESS2
        ESS2. $send(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
              pre: $ntask =$ ESS2
              eff: $active := \top$ and $ntask := \bot$

    Data Sending Session:

        DSS1. $in(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Init}}}$
              pre: $active = \top$, $smes$ and $ntask = \bot$
              eff: $smes := m$ and $ntask :=$ DSS2
        DSS2. $send(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
              pre: $m := smes$ and $ntask =$ DSS2
              eff: $smes := \bot$ and $ntask := \bot$
        DSS3. $receive(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
              pre: $rmes$ and $ntask = \bot$
              eff: $rmes := m$ and $ntask :=$ DSS4
        DSS4. $out(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Init}}}$
              pre: $m := rmes$ and $ntask =$ DSS4
              eff: $rmes$ and $ntask := \bot$

    Expire Session:

        EXS1. $in(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\overline{\text{Init}}}$
              pre: $active = \top$, $smes, rmes$ and $ntask = \bot$
              eff: $:= \bot$ and $ntask :=$ EXS2
        EXS2. $send(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
              pre: $ntask =$ EXS2
              eff: $active$ and $ntask := \bot$
```

Figure 7.54: Code for ideal Initiator of Secure Channel, $\overline{\text{Init}}_{\text{SC}}$ (Part II)

Code for ideal Receiver of Secure Channel, $\overline{\text{Rec}}_{\text{SC}}$

Signature:

$\texttt{sid}_{\text{SC}} = (\overline{\text{Init}}, \overline{\text{Rec}}, \texttt{sid}'_{\text{SC}})$

Input:	Output:
$\mathit{in}(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\overline{\text{Rec}}}$	$\mathit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
$\mathit{in}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Rec}}}$	$\mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
$\mathit{receive}(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$	$\mathit{out}(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Rec}}}$
$\mathit{in}(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\overline{\text{Rec}}}$	$\mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$

State:

$\mathit{smes}, \mathit{rmes} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$ $\mathit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$
$\mathit{active} \in \{\perp, \top\}$, initially $\perp$

Tasks:

$\{\mathit{send}(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}, \mathit{send}(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}},$
$\mathit{out}(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Rec}}}, \mathit{send}(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}\}$

Figure 7.55: Code for ideal Receiver of Secure Channel, $\overline{\text{Rec}}_{\text{SC}}$ (Part I)

Code for ideal Receiver of Secure Channel, $\overline{\text{Rec}}_{\text{SC}}$

```
Transitions:

    Establish Session:

        ESS1. $in(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\overline{\text{Rec}}}$
                pre: $active$ and $ntask = \bot$
                eff: $ntask :=$ ESS2

        ESS2. $send(\texttt{Establish}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
                pre: $ntask =$ ESS2
                eff: $active := \top$ and $ntask := \bot$

    Data Sending Session:

        DSS1. $in(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Rec}}}$
                pre: $active = \top$, $smes$ and $ntask = \bot$
                eff: $smes := m$ and $ntask :=$ DSS2

        DSS2. $send(\texttt{Send}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
                pre: $ntask =$ DSS2
                eff: $m := smes$ and $ntask := \bot$

        DSS3. $receive(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\text{F}_{\text{SC}}}$
                pre:$rmes$ and $ntask = \bot$
                eff: $rmes := m$ and $ntask :=$ DSS4

        DSS4. $out(\texttt{Receive}, \texttt{sid}_{\text{SC}}, m)_{\overline{\text{Rec}}}$
                pre: $ntask =$ DSS4
                eff: $rmes := m$ and $ntask := \bot$

    Expire Session:

        EXS1. $in(\texttt{Expire}_{\text{SC}}, \texttt{sid}_{\text{SC}})_{\overline{\text{Rec}}}$
                pre: $active = \top$, $smes, rmes$ and $ntask = \bot$
                eff: $ntask :=$ EXS2

        EXS2. $send(\texttt{Expire}_{\texttt{DIC}}, \texttt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
                pre: $ntask =$ EXS2
                eff: $active$ and $ntask := \bot$
```

Figure 7.56: Code for ideal Receiver of Secure Channel, $\overline{\text{Rec}}_{\text{SC}}$ (Part II)

Code for Simulator for Secure Channel, Sim_{SC}

Signature:

$\mathtt{sid}_{\text{SC}} = (\overline{\text{Init}}, \overline{\text{Rec}}, \mathtt{sid}'_{\text{SC}})$

Input:
$\textit{receive}(\mathtt{SID}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$
$\textit{receive}(\mathtt{Send}, \mathtt{sid}_{\text{SC}}, |m|)_{\text{F}_{\text{SC}}}$
$\textit{receive}(\mathtt{Expire}_{\text{SC}}, \mathtt{sid}_{\text{SC}})_{\text{F}_{\text{SC}}}$

Output:
$\textit{send}(\mathtt{Response}, \mathtt{sid}_{\text{SC}}, ok)_{\text{F}_{\text{SC}}}$

Other:
*Other arbitrary tasks are included the basic input/internal/output tasks such as corrupt message , *chooserand*, *rand*$(*)$ and *out*$(*)$.

State:

$\textit{active} \in \{\perp, \top\}$, initially $\perp$	$\textit{smes} \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$
$\textit{ntask} \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$	$\textit{length} \in (\{0,1\}^*) \cup \{\perp\}$, initially

Other arbitrary variables; call ”new” variables.

Tasks:

$\{\textit{send}(\mathtt{Response}, \mathtt{sid}_{\text{SC}}, ok)_{\text{F}_{\text{SC}}}\}$

Figure 7.57: Code fot Simulator for Secure Channel, Sim_{SC} (Part I)

Code for Simulator for Secure Channel, $\mathrm{Sim}_{\mathrm{SC}}$

Transitions:

Establish Session:

```
ESS1. $receive(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F_{SC}}}$
        pre: $active, ntask = \bot$
        eff: $active := \top$.
```

Data Sending Session:

```
DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{F_{SC}}}$
        pre: $active = \top$, $ntask = \bot$
        eff: $length := |m|$ and $ntask := \mathrm{DSS2}$
DSS2. $send(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, ok)_{\mathrm{F_{SC}}}$
        pre: $ntask = \mathrm{DSS2}$
        eff: $length, ntask := \bot$
```

Expire Session:

```
EXS1. $receive(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{F_{SC}}}$
        pre: $active = \top$
        eff: $active := \bot$
```

Other tasks:

This simulator makes arbitrary tasks to simulate the real world protocol system $\mathrm{Real}_{\mathrm{SC}}$. The tasks mey be run with the information obtained from the simulator. Additionaly, this simulator can output the message from the adversary of the simiulating world to the environment.

Figure 7.58: Code fot Simulator for Secure Channel, $\mathrm{Sim}_{\mathrm{SC}}$ (Part II)

Code for Random Source $Src(D,\mu)$, F_{SRC},
parameterized by probability distribution (D,μ),
where μ is the uniform distribution over distribution D from [17].

```
Signature:
    Input: none
    Output: $rand(d), d \in D$
    Internal: $chooserand$
State:
    $chosenval \in D \cup \{\perp\}$, initially $\perp$
Transitions:
    $chooserand$
        pre: $chosenval = \perp$
        eff: $chosenval :=$ choose-random$(D,\mu)$
    $rand(d)$
        pre: $d = chosenval$
        eff: none
Tasks:
    $\{chooserand, rand(*)\}$
```

Figure 7.59: Code for $Src(D,\mu)$, F_{SRC}

7.4 Equivalence of Three Cryptographic Channels

From the four above-mentioned theorems, we can immediately obtain the following main theorem.

Theorem 8. *The three channels, SC, 2AC, and DIC are reducible to each other under some specific types of schedules.*

Chapter 8

Conclusion

This thesis focused on the security of KEM and DEM for ISO, and Universal Composability for SC, 2AC, and DIC with task PIOA. The results are itemized hereafter.

1. We introduced three appropriate definitions of NM for KEM, SNM, CNM and PNM.

2. The NMs are equivalent to each other for three attack types; CPA, CCA1, and CCA2.

3. The definition of IND is equivalent to that of the NM for KEM under CCA2.

4. A protocol of KEM, Σ, UC-realizes $\mathcal{F}_{\mathrm{KEM}}$ if and only if Σ is IND-CCA2 KEM.

5. A protocol of DEM, Σ', UC-realizes $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ in the $\mathcal{F}_{\mathrm{KEM}}$ hybrid model if and only if Σ' is IND-P2-C2 DEM.

6. The three cryptographic channels, SC, 2AC, and DIC, are reducible to each other. More specifically, we showed that 2AC and DIC are reducible to each other under some types of schedule and that DIC and SC are reducible to each other under some types of schedules in the UC framework with PIOA model.

Bibliography

[1] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys," Advances in Cryptology EUROCRYPT'93, LNCS, Vol. 765, pp. 398–409, Springer Verlag, 1993.

[2] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key-exchange protocols," 30th STOC, ACM, 1998.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes," CRYPTO'98, LNCS, Vol.1462. Springer Verlag, 1998.

[4] M. Backes, B. Pfitzmann and M. Waidner, "A Universally Composable Cryptographic Library," http://eprint.iacr.org/2003/015.

[5] M. Backes, B. Pfitzmann, and M. Waidner, "A Composable Cryptographic Library with Nested Operations," 10th ACM, CCS, 2003.

[6] M. Backes, B. Pfitzmann and M. Waidner, "A General Composition Theorem for Secure Reactive Systems," 1st TCC, LNCS, Vol. 2951, pp. 336–354, 2004.

[7] M. Backes, B. Pfitzmann, and M. Waidner, "Secure Asynchronous Reactive Systems," Cryptology ePrint Archive, Report 082, 2004. http://eprint.iacr.org/.

[8] M. Bellare and P. Rogaway, “The Game-playing Technique and Its Application to Triple Encryption,” Cryptology ePrint Archive, Report 331, 2004. http://eprint. iacr.org/.

[9] M. Bellare and P. Rogaway, “Random oracles are practical: a paradigm for designing efficient protocols,” In 1st Conference on Computer and Communications Security, pp.62–73, ACM, 1993.

[10] M. Bellare and A. Sahai, “Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterisation,” CRYPTO’99, LNCS, Vol.1666, Springer Verlag, 1999.

[11] M. Ben-Or, S. Goldwasser, and A. Wigderson, “Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation,” STOC, pp.1–10, 1988.

[12] R. Canetti, “Security and composition of multi-party cryptographic protocols,” Journal of Cryptology, Vol. 13, No. 1, 2000. Available at http://philby.ucsd.edu/cryptolib/1998/98-18.html.

[13] R. Canetti, “Universally Composable Security: A New paradigm for Cryptographic Protocols,” 42nd FOCS, 2001. Full version available at http://eprint.iacr.org/2000/067.

[14] R. Canetti, “Universally Composable Signature, Certification, and Authentication,” August, 2004. http://eprint.iacr.org/2003/239/.

[15] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala, “Taskstructured probabilistic I/O automata,” WODES’06, 2006.

[16] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala, “Timebounded task-PIOAs: a framework for analyzing security protocols,” DISC’06, 2006.

[17] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala, “Using probabilistic I/O automata to analyze an oblivious transfer protocol,” Technical Report MIT-CSAIL-TR-2006-046, CSAIL, MIT, 2006. This is the revised version of Technical Reports MIT-LCS-TR-1001a and MIT-LCS-TR-1001.

[18] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala, “Using Task-Structured Probabilistic I/O Automata to Analyze an Oblivious Transfer Protocol,” This is a revised version of Technical Report MIT-CSAIL-TR-2006-046, http://eprint.iacr.org.

[19] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. “Universally Composable Security with Global Setup,” http://eprint.iacr.org/2006/432.

[20] R. Canetti and J. Herzog, “Universally Composable Symbolic Analysis of Cryptographic Protocols (The case of encryption-based mutual authentication and key exchange),” http://eprint.iacr.org/2004/334.

[21] R. Canetti and H. Krawczyk, “Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels,” EUROCRYPT’01, 2001. Full version at http://eprint.iacr.org/2001.

[22] R. Canetti and H. Krawczyk, “Universally Composable Notions of Key Exchange and Secure Channels,” EUROCRYPT’02, LNCS, Springer Verlag, 2002. http://eprint.iacr.org/2002.

[23] R. Canetti, E.Kushilevitz and Y. Lindell, “On the Limitations of Universally Composable Two-Party Computation Without Set-up Assumptions,” an extended abstract presented at EUROCRYPT’03, 2003. Available at http://eprint.iacr.org/2004/116/.

[24] R. Canetti and T. Rabin, “Universal Composition with Joint State,” Proceedings of CRYPTO’03, LNCS, Springer Verlag, 2003. Available at http://eprint.iacr.org/2002.

[25] R. Canetti and T. Rabin, "Universal Composition with Joint State," CRYPTO'03, 2003. Also http://eprint.iacr.org/2002.

[26] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty Unconditionally Secure Protocols" STOC, pp. 11–19, 1988.

[27] R. Cramer and V. Shoup, "A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack," CRYPTO '98, 1998.

[28] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," http://shoup.net/papers/, Dec. 2001.

[29] D. Dolev, C. Dwork, and M. Naor, "Non-Malleable Cryptography," 23rd STOC, 1991. Also Technical Report CS95-27, Weizmann Institute of Science, 1995.

[30] O. Goldreich, "Foundations of Cryptography (Fragments of a book)," Weizmann Inst. of Science, 1995. (Available at http://philby.ucsd.edu)

[31] O. Goldreich, "Secure Multi-Party Computation," 1998. (Available at http://philby.ucsd.edu) bibitemSS S. Goldwasser and S.Micali, "Probabilistic Encryption," Journal of Computer and System Sciences, 28: pp. 270–299, 1984.

[32] S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," SIAM, Vol. 18, No. 1, pp. 186–208, 1989.

[33] S. Goldwasser, S. Micali, and R.L. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM, pp. 281–308, Apr. 1988.

[34] J. Håstad, "Pseudo-Random Generators under Uniform Assumptions," STOC, 1990.

[35] J. Herranz, D. Hofheinz and E. Kiltz, “KEM/DEM: Necessary and Sufficient Conditions for Secure Hybrid Encryption,” IACR ePrint Archive 2006/265, http://eprint.iacr.org.

[36] R. Impagliazzo, L. Levin, and M. Luby. “Pseudo-Random Number Generation from One-Way Functions,” STOC, pp.12–24, 1989.

[37] J. Katz and M. Yung, “Characterization of Security Notions for Probabilistic Private-Key Encryption.” Full version available at http://www.cs.umd.edu/˜jkatz/.

[38] H. Krawczyk, “The order of encryption and authentication for protecting communications (Or: how secure is SSL?),” CRYPTO’01, 2001.

[39] S. Miyagawa, K. Yoneyama, K. Ohta and N. Kunihiro, “Non-Malleability for KEM and Tag-KEM Reconsidered,” SCIS’07, 4C1-1, Jan. 2007.

[40] P. Mateus, J.C. Mitchell, and A. Scedrov, “Composition of Cryptographic Protocols in a Probabilistic Polynomial-time Calculus,” CONCUR’03, Concurrency Theory, LNCS, Vol. 2761, pp. 327–349, Springer Verlag, 2003.

[41] W. Nagao, Y. Manabe and T. Okamoto, “A Universally Composable Secure Channel Based on the KEM-DEM Framework,” TCC’05, LNCS, Vol. 3378, pp. 426–444, Springer Verlag, 2005.

[42] W. Nagao, Y. Manabe and T. Okamoto, “A Universally Composable Secure Channel Based on the KEM-DEM Framework,” IEICE Transactions on Fundamentals, Vol. E89-A, pp. 28-38, Jan. 2006.

[43] W. Nagao, Y. Manabe, and T. Okamoto, “On the Equivalence of Several Security Notions of KEM and DEM,” IEICE Transactions on Fundamentals, Vol. E91-A, pp. 283–297, Jan. 2008.

[44] M. Naor, “Bit Commitment Using Pseudo-Randomness,” CRYPTO’89, LNCS, Vol. 435, pp. 128–136, Springer Verlag, 1990.

[45] M. Naor and M.Yung, “Universal One-Way Hash Functions and Their Cryptographic Applications,” STOC, pp.33–43, 1989.

[46] T. Okamoto, “On the Relationship among Cryptographic Physical Assumptions,” ISAAC’93, LNCS Vol. 762, pp. 369–378, Springer Verlag, 1993.

[47] M. Prabhakaran and A. Sahai, “New Notions of Security: Achieving Universal Composability without Trusted Setup,” manuscript, Jun. 2004. Available at: http://eprint.iacr.org/2004/139.

[48] J. Rompel, “One-Way Functions are Necessary and Sufficient for Secure Signature,” STOC, pp.387–394, 1990.

[49] R. Segala, “Modeling and verification of randomized distributed real-time systems,” Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, May, 1995. Also, MIT/LCS/TR-676.

[50] R. Segala, “Compositional Verification of Randomized Distributed Algorithms,” Compositionality LNCS, Vol. 1536, pp. 515–540, Springer Verlag, 1997.

[51] R. Segala and N. Lynch, “Probabilistic simulations for probabilistic processes,” Nordic Journal of Computing, Vol. 2, No. 2, pp. 250–273, 1995.

[52] N. Lynch, R. Segala and F. Vaandrager, “Observing branching structure through probabilistic contexts,” SIAM Journal on Computing, Vol. 37, No. 4, pp. 977–1013, 2007.

[53] V. Shoup, “On Formal Models for Secure Key Exchange,” manuscript, 1999. Available at: http://www.Shoup.org.

[54] N. Lynch, R. Segala, and F. Vaandrager, “Compositionality for Probabilistic Automata,” 14th International Conference on Concurrency Theory, LNCS, Vol. 2761, pp. 208–221, Sep, Springer Verlag, 2003. Full version in MIT-LCS-TR-907, MIT.

[55] V. Shoup, “A Proposal for an ISO Standard for Public Key Encryption (version 2.1),” ISO/IEC JTC1/SC27, N2563, http://shoup.net/papers/, Dec. 2001.

[56] V. Shoup, “Sequences of Games: A Tool for Taming Complexity in Security Proofs,” Cryptology ePrint Archive, Report 332, 2004. http://eprint.iacr.org/.

[57] S. Tarento, “Machine-checked Security Proofs of Cryptographic Signature Schemes,” ESORICS’05, 10th European Symposium on Research in Computer Security, LNCS, Vol. 3679, Springer Verlag, 2005.

[58] M. Waidner, “Unconditional Sender and Recipient Untraceability in spite of Active Attacks,” EUROCRYPT’89, LNCS Vol. 434, Springer Verlag, pp. 302–319, 1990.

[59] D. Wikstrm, “A Universally Composable Mix-Net,” TCC 2004, LNCS Vol. 2951, pp. 317–335, 2004.

[60] NESSIE, http://cryptonessie.org.

Publications

Major Publications

Journals

1. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, “A Universally Composable Secure Channel Based on the KEM-DEM Framework,” *IEICE Transactions on Fundamentals*, No. 1, Vol. E89-A,, pp. 28–38, 2006.

2. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, “On the Equivalence of Several Security Notions of KEM and DEM,” *IEICE Transactions on Fundamentals*, No.1, Vol. E91-A,, pp. 283–297, 2008.

International Conferences

1. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, “On the Security of Hybrid Public-Key Encryption,” *CITSA ’04,* Vol. 1, pp. 28–33, IIIS, 2004.

2. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, “A Universally Composable Secure Channel Based on the KEM-DEM Framework,” *TCC’05*, LNCS, Vol. 3378, pp. 426–444, Springer Verlag, 2005.

3. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, “Relation

of Three Cryptographic Channels in the UC Framework," *ProvSec 2008*, LNCS, Vol. 5324, pp. 268–282, Springer Verlag, 2008.

Other Publications

Simposiums

1. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, "On the Security of Hybrid Public-Key Encryption," *SCIS '04*, Vol. 1, pp. 35–40, 2004 (in Japanese).

2. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, "A Universally Composable Secure Channel Based on the KEM-DEM Framework," *SCIS '05*, Vol. 1, pp. 163–168, 2005.

Cryptology ePrint Archive

1. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto, "On the Equivalence of Several Security Notions of Key Encapsulation Mechanism," Cryptology ePrint Archive, No. 2006/268, http://eprint.iacr.org/2006/268.pdf, 2006.