Master Thesis

Design and Implementation of a Certified Mail Exchange System Using Simultaneous Secret Exchange

Supervisor Professor Yasuo OKABE

Department of Intelligence Science and Technology
Graduate School of Informatics
Kyoto University

Keita SHIMIZU

February 6, 2009

Design and Implementation of a Certified Mail Exchange System Using Simultaneous Secret Exchange

Keita SHIMIZU

Abstract

In this paper we design a certified e-mail exchange system based on simultaneous secret exchange protocol proposed by Okamoto and Ohta. We selected their protocol because it is superior in efficiency, especially at the view point of the number of sessions.

At first, we designed the whole procedure mainly based on Okamoto and Ohta's protocols, adding processes to confirm the contents each party have received. In this system, we want to realize good properties often referred in certified mail researches with TTPs, for example 'send-and-forget'. So we assume two players, an MTA (Mail Transfer Agent, e.g. SMTP Server) and an MUA (Mail User Agent, e.g. the client the sender or the receiver use) in each party, and designed construction which enables 'send-and-forget' for the sender when the sender can control an MTA, or enables to exchange messages for the sender with the receiver directly when cannot.

Then, we made message formats in the form of XML to send each transactions. We selected the method of attaching transaction XML to e-mail to communicate messages to realize reachability. With this method, it takes rather long time to complete exchanging secret keys and get contents of the mail or receipt when the mail delay is large. Then we proposed a method to reduce the number of exchanging transactions. This method doubles the number of bits sent at once after some bits have been sent regularly. The 'fairness' is defined as the maximum of the difference of time needed to calculate the remaining bits of keys at any time, where the difference is taken between two players. This amount is 'at most twice' in a normal simultaneous secret exchange protocols, while it is 'at most constant difference' in our method.

同時秘密交換を利用した配達内容証明が可能な電子メールシステムの設計および実装

清水 敬太

内容梗概

本論文で我々は証明付き電子メール配達システムを同時秘密交換プロトコルに基づいて設計した. 我々は効率, 特にセッションの回数の観点から優れていることから岡本・太田らの提案したプロトコルを選択した.

岡本らのプロトコルを主軸として, 我々は双方が受け取った内容に相異がないかの最終確認のプロセスなどを加えて, メッセージと領収書を交換するための手続きを設計した. システムを設計するにあたって我々は, TTP を用いて構成される証明付きメールの研究で頻繁に言及される Send-and-forget など良い性質を実現させたかった. そこで, 我々はそれぞれのパーティ内に MTA (Mail Transfer Agent, SMTP サーバなど) , MUA (Mail User Agent, 送信者, 受信者が用いるメールクライアント) の 2 者が存在することを仮定した. これにより送信者が MTA を制御下に置ける場合に Send-and-forget を MUA のユーザレベルでは実現でき, そうでない場合には受信者と直接メッセージ交換を行える構成を設計することができた. そして, それぞれのトランザクションを伝送するためのメッセージフォーマットを XML としてまとめた.

我々は到達可能性を考慮してメッセージの伝達手段としてメールにトランザクションの XML を添付するという形式を選んだが, この方法ではメールの一往復あたり遅延が大きい場合, 必要な秘密鍵の交換を完了してメールや領収書の内容を入手するまでにかなり長い時間がかかってしまう. そこでセッションの往復回数を減らすための手段として, ある数のビットを普通にに送った後には 1 度に送信する数を増加させていくという手法を提案した. 同時秘密交換プロトコルでは両者が鍵を計算して解こうとしたときにかかる時間の公平性が求められ, それは通常は「かかる時間割合が二倍以内である」という条件により保障されている. 我々の手法においては,「かかる時間差が大きくなることはない」ことにより保障される.

Design and Implementation of a Certified Mail Exchange System Using Simultaneous Secret Exchange

Contents

Chapter 1 Introduction

1.1 Background

We contracts signings or important messages with certified mail by off-line, as the conception of e-mail was designed to be used only among academic researchers.

Although e-mail have become the most popular way of exchanging messages above all, today we cannot make sure that an e-mail has arrived at the hand of receiver. Some methods which sender get the receiving notification have been proposed. The HTML headers 'Return-Receipt-To', or 'Disposition-Notification-To' are the examples of methods. The former is a original extension of 'send-mail', and the latter is defined in RFC 2298 [1]. With these protocols, the sender can obtain receipt if the receiver agrees to send it. However these methods fundamentally depends on receiver's cooperations. If the receiver does not want to cooperate or prevent sending the receipt, the sender cannot receive it.

So, fairness becomes the most serious problem in such situations. Against such a backgrounds, "certified e-mail" started to be studied. Researches of this realm started as a variety of the contract signing problem, but today various kinds of methods of implementations are proposed.

Today, some commercial services have launched, and the frame is standardized in ISO 13888 [2], and USPS (United States Postal Service) also started a service "Certified Mail" from 2001, which is an e-mail version of a traditional certified mail on paper mail. Today, information technology has developed greatly, and many business communications are on-line. So, the importance of this realm has been and will be increasing and more convenient alternative is required.

1.2 Structure of Thesis

This paper consists of six sections. At the beginning, we remark the background of this research and make clear what we would do in this Chapter 1.

We would explain the related works and current problems in Chapter 2. On the basis of those problems, we show an assumption and problems this thesis would concern.

Based on the assumptions we assumed in Chapter 2, we consider how to realize

the protocol in the Chapter 3, and explain how we should implement it in Chapter 4.

We show a improvement scheme for more rapid exchange in Chapter 5. In the last chapter, we summarize the contributions of this paper.

1.3 Result of This Paper

In this paper, we designed a system that includes sender clients and receiver clients, and their representational servers. They process the gradual secret exchange protocol and realize certified e-mail without TTPs. To realize them, we considered how they should process e-mail exchanges, and be implemented in a system. In this process, we designed the entire procedures and error processes, and transactions needed.

We realized ‘send-and-forget’ by considering an MUA and an MTA in each parties. At last, we considered a method to decrease the number of sessions and needed time to complete the procedure significantly.

Chapter 2 Preliminaries

2.1 Related Works

As we have already mentioned in Section 1.1, there have been already many researches about certified mail, and concerning it, contract signing. Researches about certified mail derives from those about contract singing. So in this subsection, we sometimes refer to researches about contract signing. Those researches seem to be classified by the dependency on the third party.

One type of studies uses a **"Trusted Third Party" (TTP)**, which serves as an intermediary. This type is the most major and some researches are commercialized. Merkle. R is a pioneer of this realm[Mer78], but their system was inefficient in the number of communication rounds and computation. After that, many researchers have been engaged in this category and proposed some variants. Zhou and Gollman is a representative example of researches on this model [3]. And then, Other researchers also proposed some good properties. For example, Asokan et al. proposed "optimistic protocol", in which TTP is only used in case of dispute [4]. (On the other hand, a system TTP is always used is called "on-line"), and Imamoto et al. proposed a scalable on-line system [5]. But systems on this model are fundamentally dependent on third party's reliability. So TTPs need to be managed by reliable community or company, and the burden tends to be large.

Another type of studies assumes the existence of the third party, which serves as a reliable source of randomness and can play a roll of judge [6], [7].

Although these approaches have the advantage that they make no assumptions regarding the computing power of parties, requiring the use of a third party is still considered some what strong. Moreover, this type is applied only to contract contract signing, not to certified mail.

And the other type realizes certified mail without third parties. No other party would have relation with sender and receiver in exchanging mail or contract signing, so the sender or the receiver do not need to prepare trusted third parties. They should draw on others only in case of dispute, and they can claim his or her legality if s/he

executed procedure fairly. In spite of these advantages, there are not many studies about systems on this model by comparison to certified mail systems with TTPs. Researches about this model concentrate on secret exchange protocols, especially gradual secret exchange protocol. We explain this protocol in the later subsection.

M.Luby[8] made the basic concept of "simultaneous secret exchange." They develop a slightly biased coin, and proposed the way to exchange a secret bit according to the result of flipping it. This method used the probability and law of great numbers. M.Blum [9] proposed a concept of gradual secret exchange protocol and designed a protocol based on the factoring problem. Afterward, some researches have been done on this protocol, for example Yao et al. citeyao1986gae and Cleve et al. [10]. Even, Goldreich and Lempel's protocols [11] are based on "oblivious transfer." Goldreich also referred to use it certified e-mail. Okamoto and Ohta[12] also proposed some simultaneous secret exchange protocols. Protocols they proposed are practically efficient and are proven to be secure under general assumptions such as the existence of one-way permutations and one-way functions.

In spite of these theoretical heritages, there are almost no researches which designed e-mail exchange systems based on these theories.

2.2 Basic Concept

2.2.1 Certified Mail

In the real world, there are both "contents-certified mail" and "delivery-certified mail" on paper mail. Usually the former requires greater care and includes the function of the latter. But both of them are not so different on e-mail from this viewpoint of technical difficulties. So we say "certified mail" to indicate "contents-certified mail" if not otherwise specified.

2.2.2 Party

We use this word to indicate each side of sender and receiver (Alice and Bob). If an MTA (e.g. SMTP server) is under his/her control or be trusted by him/her, we consider the MTA is in his/her the party. This discussion would concern the discussion in Section 3.2 "Structures."

2.2.3 Trusted Third Party

The word “third party” indicates a party other than the sender and receiver. Especially “trusted third party(TTP)” is a third party that both sender and receiver believe to be fair and correct, and give approval to see contents of mails.

2.2.4 Simultaneous Secret Exchange

A simultaneous secret exchange protocol is executed between a sender (we call her Alice) and receiver (we call him Bob) as follows: At first, Alice generates a secret key a (n bits string) and Bob also generates a secret key b(n bits string). We assume that these keys are secrets to each other. Then, they exchange these keys each other. To prevent a party from obtaining a secret without revealing his/her own secret, they exchange $f(a)$ and $g(b)$, here Alice cannot obtain b from $g(b)$, similarly Bob cannot obtain a from $f(a)$. Then, Alice and Bob open a and b bit by bit. Okamoto and Ohta[1994] pointed out a simultaneous secret exchange protocol needs to satisfy the following conditions.

- **Correctness** Each party can check the validity of each bit at each stage, to ensure that a garbage has not been received. If they cannot check at each stage, garbage bits can be detected finally after all n bits are sent. However, this is too late. In this occasion, dishonest party have gotten correct bits, while honest party gets garbage bits and he cannot obtain the true bits thereafter.
- **Fairness** Let $T(i)$ be the time of computing the remaining i bits when Bob received first $(n-i)$ bits of a. The difference between $T(i)$ and $T(i-1)$ $(1 \leq i \leq n)$ should be small. If the difference is non-negligible, then Bob has non-negligible advantage over Alice at the stage that Alice has released $(n-i+1)$ bits and Bob has released $(n-i)$bits. Here, note that this condition should be satisfied even when $i = 1$. That is $T(1)$ should be almost 0, since $T(0) = 0$. So, for example, the following naive protocol does not satisfy the fairness, since there exists a big gap between $T(1)$ and $T(0)$: $f(a) = (f_0(x_1), ..., f_0(x_n))$, where $a = (HC(x_1), ..., HC(x_n))$. ($HC(x)$ denotes the hard core bit of Goldreich[1989] of x) $x_i (i = 1, ..., n)$ is revealed Alice’s (or Bob’s) i-th step of the revealing phase. Then, $T(1)$ is the running time of inverting f_0, which is non-negligibly greater than $T(0) = 0$.

If these conditions are satisfied, for polynomial time machines Alice and Bob, at any stage of protocol, Bob obtains a, if and only if Alice obtains b.

Here, we assume the case Alice (or Bob) wants to send a pseudo key. If she made X from the genuine secret key and began to send the pseudo bits from the middle of secret exchange, then Bob can detect the unfairness at once. If she made X' from a pseudo key from the beginning, Bob cannot detect the unfairness until he gets the pseudo key and fails to decrypt the message. But in this case, he can at least protest that Alice sent a worthless key by showing the encrypted message and the key, and then, receipt is repealed.

Comparing simultaneous secret exchange to simple bit-by-bit exchange like this way, the former has the advantage of preventing a party from attacking like "send a different key and repeat a different message." This is not easy, but possible, especially in the case that the message is very short. If we use a simultaneous secret exchange, the pair of 'the encrypted message' and 'the secret key' are firmly fastened at the time Alice sends the first mail. So Bob can make a receipt for the pair in relief. We summarized these comparisons in Table 1.

In addition, all communication between Alice and Bob have each party's signatures. So if someone try its to attack with man-in-the-middle method, he can intercept and block the message, but cannot modify it. Replay attack would work at some level, but the time stamps can limit the efficiency.

In this realm, the name "gradual secret exchange" is often used in the papers. Typically, "gradual secret" exchange means one-directional secret releasing (so, above procedure itself represents "gradual secret exchange"), and protocols are called "simultaneous secret exchange protocol" when "gradual secret exchange protocol" is symmetrically used to exchange secrets bi-directional. Moreover, sometimes these "secret exchange" is called "secret releasing." These are seems in the same meaning. We standardize "secret exchange" throughout this paper.

2.2.5 Okamoto and Ohta'94

Okamoto and Ohta[12] proposed some patterns of simultaneous secret exchange protocols. One of protocols in their thesis, which use one-way permutations is realized as follows. We assume a problem that the sender Alice wants to send secret n-bits

Table 1: Comparison of Each Methods

	abscond	regenerate different message attack	pseudo key sending
simple sending	× Cannot Prevent	× After receiving the first message, Sender can send another key arbitrarily.	△ Cannot Prevent, but protest when the receiver cannot decrypt the message with the key.
bit-by-bit exchange	○ Preventable		
simultaneous secret exchange	○ Preventable	○ determined when first message received	

strings $s_A = (x_n, ..., x_1)$ ($s_A \in_U \{0,1\}^n$) to the receiver Bob.

Step 1 Let $\mathcal{F}$ be a family of tight one-way permutations, and $f_{\delta_i} \in \mathcal{F}$, where i denotes the security parameter.

$$f_{\delta_i} : \{0,1\}^i \to \{0,1\}^i, (i = 1, 2, ..., n)$$

Here $\mathcal{F}$ is consist of $\mathcal{F} = \{f_{\delta_1}, f_{\delta_2}, .., f_{\delta_n}\}$. f_{δ_i} is uniquely determined with δ_i.
The sender Alice randomly generates the parameters of $\mathcal{F}$, $\delta_1, \delta_2, ..., \delta_n$. Then Alice calculates n-bit string X as follows:

$$x_1^* = x_1,$$

$$x_i^* = x_i || f_{\delta_{i-1}}(x_{i-1}^*), (i = 2, ..., n),$$

$$X = f_{\delta_n}(x_n^*).$$

Here, $x||y$ means concatenation of string x and string y.

Step 2 Alice sends $(\delta_1, \delta_2, ..., \delta_n)$ and X to Bob to commit to Alice's secret s_A.

Step 3 When Bob receives them, Bob checks whether $(\delta_1, \delta_2, ..., \delta_n)$ are valid for the parameters of $\mathcal{F}$, and whether $X \in \{0,1\}^n$. If they do not hold, Bob halts the protocol. Otherwise, Bob writes $(\delta_1, \delta_2, ..., \delta_n)$, X on the output to keep them.

[End of commitiment stage]

Step 4 In descending order for $i = n, ..., 1$, repeat the following procedures sequentially. Alice sends x_i^* to Bob.

Step 5 When $i = n$, Bob checks whether $X = f_{\delta_n}(x_n^*)$ holds or not.

When $i = n - 1, ..., 1$, Bob checks whether

$$[x_{i+1}^*]_i = f_{\delta_i}(x_i^*).$$

Here, $[x]_a$ denotes the least a bits of x, and $[x]^a$ denotes the most a bits of x. If it does not hold, Bob halts the protocol. Otherwise, Bob writes x_i and $(x_i)^*$ as an output.

At last, Bob can get x_i from each x_i^* for all i (since $x_i = [x_i^*]^1$).

[End of secret releasing stage]

In using this procedure to simultaneous secret exchange, both Alice and Bob execute above protocol bi-directionally.

Protocols they proposed can be constructed based on more general assumptions such as one-way permutations and one-way functions, while the existing efficient simultaneous secret exchange protocols are based on more constrained assumptions such as specific number theoretic problem, the existence of oblivious transfer primitive or trap-door one-way permutations.

Furthermore, this protocol is one of the most efficient protocol in the protocols ever proposed in an aspect of number of sessions between Alice and Bob. We suppose Even's protocol[11], which uses oblivious transfer. To exchange key of $2n$ bits, $2 * n$ (times) oblivious transfers and $2 * 2n$ (times) normal transfer, then for all $6n$ (times) of transfers are needed. In contrast, Okamoto and Ohta's protocol needs only $4n + 2$ (times) of normal transfers to exchange keys of $2n$ bits.

This property is appreciably significant in applying the protocol to the mail exchange. Regarding these advantages, we use their protocols we explained above in our system.

2.3 Problems and Assumptions

Zhou and Gollman pointed out two problems about certified e-mails without TTPs[3].

One is about "selective receipt" they call. When the receiver received the message, he can select not to generate a receipt and send back it to the sender. To avoid

absconding with the important information, the sender usually sends the message with encrypting at first. Indeed the sender can prevent the with this way, he can still select not to send the receipt when the message is inconvenient to the receiver. Selective receipt fundamentally cannot be prevented without TTPs.

As the second problem, they pointed the unfair modification of timestamps. This means the receiver can generate a receipt, and then a time stamp included in the receipt at arbitrary times. Because of these problems, they claim that TTPs are essential for a certified mail system.

Despite these indications, we still think that certified mail systems are useful even if it cannot prevent selective receipt. For the first problem, we assume an example problem as follows.

> Alice have a business secret which enables making big money, and she wants to sell it to Bob. She needs a proof that Bob received the secret if Bob received the secret. Bob also wants a proof that Alice sends a genuine secret, or he claims that she sent a fake secret if the received secret is not genuine.

In the case of the above example selective receipt would not be a problem. So we make an assumption that both sender and receiver basically agree to exchange a mail. Both parties hope to exchange secret and receipt fairly, and a party can protest to others(courts for example) only when the other party act a dishonest.

For the second problem, we think the first message and other parts of procedures should possess an expiration time. With this configuration, each party needs to return a reply in some times, or the procedure would fail and the exchanging would stop. Though we don't use one in this paper, if a system uses a third party for time-stamp, we can expect more time accuracy for the system.

Under these assumptions, we think a system without TTPs would be helpful to enable to exchange secret fairly. We think our system would resolve following problems.

1. Alice sent a message and Bob received it. But Bob claims that he did not receive it.

2. Alice did not send a message to Bob. But she claims that she has certainly sent it to Bob.
3. Alice sent a fake message to Bob, and Bob received it. But Alice claims that she sent the genuine one to Bob.

We consider how the system resolves these problems in Section 5.1 “Consideration.”

At last, we add an assumption about transmission channel. Alice’s MUA and Bob’s MUA can always communicate with e-mail, but not always with other methods. This is because there are often barriers blocks their communications, e.g. firewalls.

Chapter 3 Design

In this section, we show how we designed the system. We drew upon HTTP1.1[13] and some books[14][15].

3.1 Outlines of Normal Case

We assume a sender(Alice) and a receiver(Bob). Each party may be a user which use a terminal machine of network, or may be a user and an MTA which is under control of user, as previously mentioned.

The normal procedure is as follows. We assume all session is exchanged with signature.

Step 1 At first, Alice makes a message M, and write an abstract of M. And she generates a secret key K_a, and encrypt M with K_a. (We call encrypted M as $E_{k_a}(M)$.) Alice sends $E_{k_a}(M)$, X and P_a to Bob. X is a transformed K_a with Okamoto and Ohta[12], and P_a, the parameters of it.

Step 2 When Bob receive the first message from alice, Bob checks the message validity and looks the abstract of the contents. If the abstract meets his wishes, makes a receipt token (R) contains the abstract and his signature, and dates. Then he generate a secret key K_b, encrypt R with it($E_{k_b}(R)$). Then he also generate Y, a transformed K_b with Okamoto and Ohta[1994] and P_b. He sends $E_{k_b}(R)$, Y, and P_b to Alice.

If Alice can decode $E_{k_b}(R)$ and gets R, she can protest Bob certainly received the Message. Here, K_a and K_b need to be the same encryption method and to have the same length.

Step 3 Alice and Bob execute the simultaneous secret exchange to exchange K_a and K_b, referring to X, Y and P_a, P_b. The exchange begins from Alice, and each party exchange keys bit by bit.

Step 4 When all bits of K_a is sent, Alice decrypts $E_{k_b}(R)$ with K_b and gets R. Bob gets M same as Alice.

Step 5 Each of them confirm the contents. Alice makes a digest of M, $H(R)$. She sends $H(R)$ to Bob.

Step 6 Bob received $H(R)$. If $H(R)$ was correct, he also makes $H(M)$ and sends it

to Alice.

Step 7 Alice confirm $H(M)$ and it is correct, this procedure ends.

We explain about error cases later Section 3.5.

3.2 Structure

Here, we assume parties which are consist of a user using an MUA and an MTA the user controls. Each user trusts on the MTA, and can delegate some part of procedures.

On such assumption, we consider how these should members acts in executing procedures. Considering the rolls of each members, we suppose regal patterns can be classified into following 3 cases.

- **Direct Style** Sender Alice and Receiver Bob exchanges secret each other directly.
- **Agent Style** Alice delegates secret key and secret exchange procedure to the MTA when she sends the first mail. And Bob received the mail, he also delegates secret key and procedure to the MTA of his party. Then the MTAs of both sides exchange the secret key, and both users receive the message or receipt.
- **Half Agent Style** Alice delegates secret exchange procedure to the MTA when she sends the first mail. But Bob does not delegate. After Bob received the mail, he exchange secrets with the MTA of Alice's party.

In what follows, we would check up the advantages and disadvantages of each style.

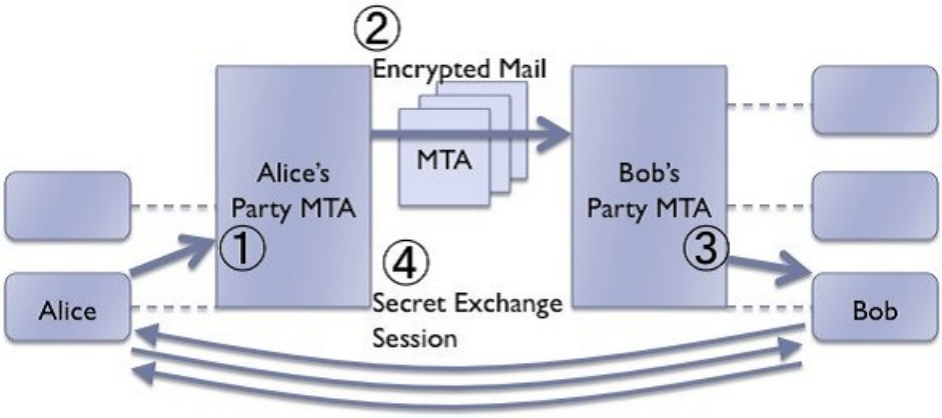
3.2.1 Direct Style

User Alice and Bob exchanges secrets each other directly. In this style, the whole session are closed in **end-to-end**, a great advantage to be introduced, since Alice and Bob need not to have control of an MTA to have session each other.

Pointing disadvantage, Alice must wait for Bob's response. So **Send-and-Forget** cannot be realized in this style. "Send-and-Forget" is a good manner which is often discussed about around the realm of certified mail, which represent "sender need not wait for session after send the first message, and all s/he needs to do for getting receipt is only just waiting." So with this model, burden onto Alice is significantly large.

In addition, it sometimes can be difficult to have session by the methods except

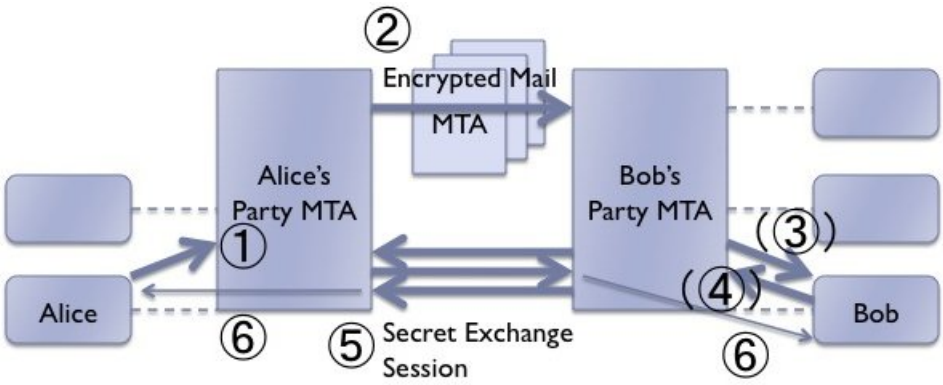
Figure 1: Direct Style



e-mail, since there can be firewalls, NAT, and so on.

3.2.2 Agent Style

Figure 2: Agent Style

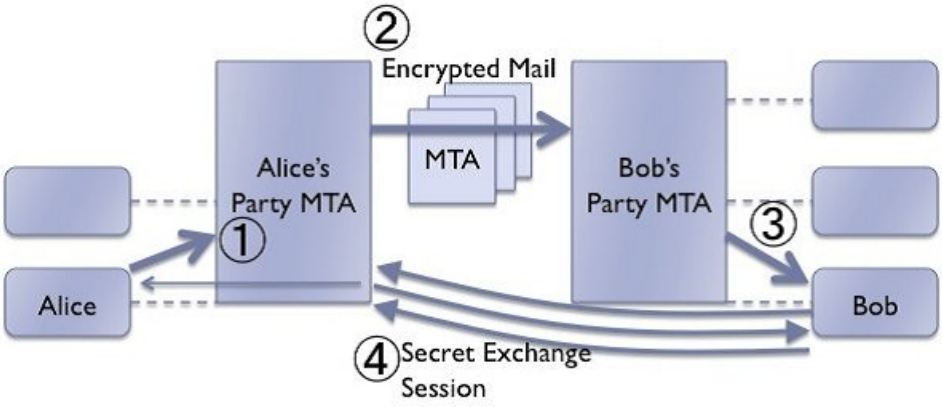


User Alice send the first mail to Bob through the MTA she trusts, and the MTA represent her and exchange secrets as her agent. After Bob received the mail, he also delegates to the MTA of his party. Then, each agent MTAs exchange the secret keys.

As an advantage, “Send-and-Forget” is realized with this style,.

As a disadvantage, introduction cost is large comparing to direct style. Both Alice and Bob need to introduce the system on their MTA.

Figure 3: Half Agent Style



3.2.3 Half Agent Style

User Alice send the first mail to Bob through the MTA she trusts, and the MTA represent and exchange secrets as her agent. After Bob received the mail, he exchange with Alice's agent MTA. In this style, **Send-and-Forget** is realized. Introduction is easier than agent style. In addition it would be easier for Bob to have session with an MTA by the methods except e-mail.

Considering disadvantage, it would be difficult for Bob to reply as a certified mail as this style is asymmetrical. We think this would not be so many case, considering the way of certified mail. To say more, if both side introduce the system of this style, users can exchange secrets bi-directionally.

Considering these advantages and disadvantages, we decided to construct the system realizes 'Direct Style' and 'Half Agent Style', and Alice can select how to use. So it is desirable that Alice can use the agent with minimum difference from the way of using in "Direct Style."

3.3 Transaction Method

We need to design how Alice and Bob communicate and send transactions to each other. Here, we made an assumption about transmission channel as follows.

> Alice's MUA and Bob's MUA can always communicate with e-mail, but not always with other methods. This is because there are often barriers

blocks their communications, e.g. firewalls.

We made decision to realize both ‘Direct Style’ and ‘Half Agent Style’ in the previous section. We want to ensure the reachability always for Alice and Bob, so we adopt e-mails for the communication method.

We made transaction message body with XML. XML is superior in the points as follows.

- It is easy to be used in Internet environment. it is constructed on text data, so suitable for e-mail.
- It can represent common computer science data structures: records, lists and trees.
- The hierarchical structure is suitable for this application.
- It has strict syntax and parsing requirements, so easy to be verified.
- In can contains binary data with base 64 encoding, which is also used for MIME contents.

We defined message formats on XML in Section 4.1. Parties attach this XML message to e-mails and send it.

3.4 Session Management

3.4.1 Mail ID and Session ID

In this system, each certified e-mail session has unique id (we call ‘MID’, an abbreviated expression of ‘mail id’). Each certified e-mail session would execute a simultaneous secret exchange session, which contains two gradual secret exchange protocol. Then we allot unique ids for each gradual secret exchange session (we call ‘SID’, an abbreviated expression of ‘session id’).

The MID and the first SID is generated by the sender Alice and stored her client. If she use the agent MTA, it stores the MID and the SID.

3.4.2 Expiration Time

When Bob received the message from Alice or her agent, Bob stores SID and MID and generates another SID and store it. Here, there are one MID and two SIDs. He sends a SID he generated with the MID in his encrypted receipt.

Alice, or her agent, who received the encrypted receipt stores the another SID.

When each SID generated, it can have a expiration time optionally. Each player stores the expiration time with SID.

In case one of expire time comes before they complete the secret exchange, s/he must report time out error. We think about errors at next subsection.

3.5 Error Case

When processing procedures, various kinds of errors are supposed to occur. We classified errors into three categories, and allot error codes for each kind of errors. If an error occurs, the player should inform it to the other party. S/he use the xml format at the next subsection along with error codes to inform what happened at his/her hands.

Error codes are as follows.

Protocol Errors The message is invalid, so parties should not process it.

- **400 Bad Request**: If the message is invalid because of wrong or lack of message elements, this error is reported.
- **412 Request Too Large**: If the message is too large, this error is reported.
- **416 Contents Invalid**: If the contents (encrypted message, date, value) is invalid, this error is reported.

Processing Errors Failures and halts occur in processing. This includes user-oriented ones.

- **408 Time Expired**: When the expired time is overred before complete secret exchanges, this error is reported.
- **481 Call/Transaction Not exist**: When a message which does not correspond to current procedures arrives, this error is reported.
- **486 Now Busy**: If the application system cannot carry on the procedure because of some problems or burdens, this error is reported.

User Errors User-oriented halts, includes not much of secret exchange checking.

- **488: Not Acceptable**: If the receiver Bob does not want to receive the message, he can select reporting this error and end the procedure.
- **489: Value Not Match**: If the value does not have consistency while processing secret exchange, this error is reported.
- **430: Terminated**: When a party want or need to stop the procedure from

some reasons, s/he can use this error.

Restart Session some error occur even if the transaction has no fault. The party received the error report can select to restart sessions if the reported error is ‘408: Time Expired’ or ‘486 Now Busy’. If s/he want to restart the session, s/he should behave as following.

- In case 408, s/he can regenerate new sessions with new SID under the same MID. To regenerate sessions, s/he should use the transaction xml of ⟨restart/⟩.
- In case 486, s/he can restart the session. S/he need to resend the same transaction at regular intervals.

In addition, if s/he cannot receive the responding transaction, s/he can resend the same transaction to the other. If a party receive the same message twice or more, need to ignore it, not to report ‘481 Call/Transaction Not exist’. This process need to avoid the case each party would wait for each other forever, as it can be happen the transaction mail disappears somewhere.

Chapter 4 Implementation

4.1 Message Format

We made a message format which is consist of xml list.

We show a example of message format, which the sender Alice sends to Bob at the Step (1).

Encrypted Message

```
<send>
    <mid>83f93ac93dabd342</mid>
    <sid>f39aac353d2f0235</sid>
    <from>alice@i.kyoto-u.ac.jp</from>
    <to>bob@i.net.ist.kyoto-u.ac.jp</to>
    <mail-dest>alice@cmtp.net.ist.i.kyoto-u.ac.jp
    </mail-dest>
    <date>20011101184732</date>
    <expire>20011102184732</expire>
    <abst>(abstract of message)</abst>
    <c-mes>(encrypted message($E_{k_a}(M)$))</c-mes>
    <key type="DES" length="64"/>
    <protocol type="okamoto">
      <envelope>(transformed secret key($X$))</envelope>
      <params>
        <param num="1">(parameter 1)</param>
        <param num="2">(parameter 2)</param>
        <param num="3">(parameter 3)</param>
        <param num="4">(parameter 4)</param>
        <param num="5">(parameter 5)</param>
        <param num="6">(parameter 6)</param>
        ...
        <param num="64">(parameter64)</param>
      </params>
       </protocol>
</send>
```

- **mid** Unique ID provided for each Certified Mail.
- **sid** Unique ID provided for each gradual secret exchange session. Two **sid** correspond to one **mid**.
- **from** Mail address of message sender.
- **to** Mail address of message receiver.
- **mail-dest** Mail address when receiver wish to exchange secrets on e-mail. This element must be applied.
- **date** Time Stamp. Recorded in a format “YYYYMMDDttmmss.”

- **expire** Expiration time. This time corresponds to SID.
- **abst** Abstract of message. This message is written in main body of e-mail in the same way.
- **c-msg** Encrypted message. This element is written in Base64 format. This element corresponds to $E_{k_a}(M)$.
- **key** The format type and length of secret key used to encrypt message. Each is described in attributes of element.
- **protocol** Appoint the protocol type used simultaneous secret exchange. (At the moment, only "okamoto" is supported.)
- **envelope** Transformed secret key in Okamoto's protocol, which works as a verification bits in secret exchange. Corresponds to X.
- **param** Parameters used to determine one-way permutation in Okamoto's protocol. This corresponds to P_a.

Contents of Message

```
<mssg>
  <from>Alice</from>
  <to>Bob</to>
  <date>20011101184732</date>
  <message>(Main Contents)</message>
</mssg>
```

- **from** Name of message sender.
- **to** Name of message receiver.
- **date** Time Stamp. Recorded in a format "YYYYMMDDttmmss."
- **message** Main Message.

Encrypted Receipt Receiver Sends to Sender

```
<rcv>
  <mid>83f93ac93dabd342</mid>
  <sid>df830a82bd8f3034</sid>
  <from>bob@i.net.ist.kyoto-u.ac.jp</from>
  <to>alice@i.kyoto-u.ac.jp</to>
  <date>20011101194732</date>
  <expire>20011103194732</expire>
    <abst>(abstract of message)</abst>
  <c-rcpt>(encrypted receipt)</c-rcpt>
  <key type="DES" length="64"/>
```

```
  <protocol type="okamoto">
    <envelope>(transformed receipt($Y$))<envelope>
    <params>
      <param num="1">(parameter 1)</param>
      <param num="2">(parameter 2)</param>
      <param num="3">(parameter 3)</param>
      <param num="4">(parameter 4)</param>
      <param num="5">(parameter 5)</param>
      <param num="6">(parameter 6)</param>
      ...
      <param num="64">(parameter64)</param>
    </params>
  </protocol>
</rcv>
```

- **mid** Unique ID provided for each Certified Mail.
- **sid** Unique ID provided for each gradual secret exchange session. Two **sid** correspond to one **mid**.
- **from** Mail address of receipt the sender(message receiver).
- **to** Mail address of receipt the receiver(message sender).
- **date** Time Stamp. Recorded in a format "YYYYMMDDttmmss"
- **expire** Expiration time. This time corresponds to SID.
- **abst** Abstract of message received at the previous step.
- **c-rcpt** Encrypted receipt. This element is written in Base64 format. This element corresponds to $E_{k_b}(R)$.
- **key** The format type and length of secret key used to encrypt receipt. Each is described in attributes of element.
- **protocol** Appoint the protocol type which is same to the one sender appointed used simultaneous secret exchange. (At the moment, only "okamoto" is supported.)
- **envelope** Transformed secret key in Okamoto's protocol, which works as a verification bits in secret exchange. Corresponds to Y.
- **param** Parameters used to determine one-way permutation in Okamoto's protocol. This corresponds to P_b.

Contents of Receipt

```
<rcpt>
  <from>Bob</from>
```

```
  <to>Alice</to>
  <date>20011101184732</date>
  <commitment>"I agree I received the message
               in the envelope."</commitment>
  <c-msg>(encrypt message received at step (1))</c-msg>
  <envelope>(transformed secret key)</envelope>
  <params>
    <param num="1">(parameter 1)</param>
    <param num="2">(parameter 2)</param>
    <param num="3">(parameter 3)</param>
    <param num="4">(parameter 4)</param>
    <param num="5">(parameter 5)</param>
    <param num="6">(parameter 6)</param>
    ...
    <param num="64">(parameter64)</param>
  </params>
</rcpt>
```

- **from** Name of receipt sender.
- **to** Name of receipt receiver.
- **date** Time Stamp. Recorded in a format "YYYYMMDDttmmss."
- **commitment** A commitment that the receiver certainly received the message encrypted with the secret key, which was transformed by the following parameters.
- **envelope** Transformed secret key in Okamoto's protocol received at Step(1).
- **c-msg** Encrypted message received at Step(1). This element is written in Base64 format. This element corresponds to $E_{k_a}(M)$.
- **param** Parameters received at Step(1).

Meesage for Each Step of Secret Exchange

```
<step>
  <mid>83f93ac93dabd342</mid>
  <sid>f39aac353d2f0235</sid>
  <value num="2">(transformed secret key at this step )</value>
  <prev>(transformed secret key received at the previous step)</prev>
</step>
```

- **mid** Unique ID provided for each Certified Mail.
- **sid** The unique id of executing certified email exchange.
- **value** The transformed secret key at each of transformation. The step number is described as an attribute.

- **prev** The transformed secret key received at the previous step. If it is the first step, this element is sent with empty.

Acknowlegement of Ending of the Session

```
<s-end>
  <mid>83f93ac93dabd342</mid>
  <sid>f39aac353d2f0235</sid>
</s-end>
```

- **mid** Unique ID provided of the Certified Mail.
- **sid** Unique ID provided for each gradual secret exchange session.

Certification from Receiver to Sender

```
<confirm>
  <mid>83f93ac93dabd342</mid>
  <sid>f39aac353d2f0235</sid>
  <date>20011101184732</date>
  <m-dgst>(digest of received message)<m-dgst>
</confirm>
```

- **mid** Unique ID provided to the Certified Mail.
- **date** Time stamp.
- **m-dgst** Digest of received message. This digest is generate by applying SHA-1 to received message. Described in base64.

Response to Certification

```
<confirm_rep>
  <mid>83f93ac93dabd342</mid>
  <date>20011101184732</date>
  <m-dgst>(digest)<m-dgst>
  <reply>ok</reply>
</confirm_rep>
```

- **mid** Unique ID provided to the Certified Mail.
- **date** Time Stamp. Recorded in a format “YYYYMMDDttmmss.”
- **m-dgst** The message digest received at the previous step.
- **reply** Reply the message digest was correct or not with “ok” or “no.”

Session End Request from message receiver to sender

```
<endreq>
  <mid>83f93ac93dabd342</mid>
</endreq>
```

- **mid** Unique ID provided to the Certified Mail.

Reply to Session End Request

```
<endrep>
  <mid>83f93ac93dabd342</mid>
</endrep>
```

- **mid** Unique ID provided to the Certified Mail.

Error Report

```
<error>
  <mid>83f93ac93dabd342</mid>
  <code>400</code>
  <target>
    <confirm>
      <mid>83f93ac93dabd342</mid>
      <sid>f39aac353d2f0235</sid>
      <date>20011101184732</date>
      <m-dgst>(digest of received message)<m-dgst>
    </confirm>
  </target>
</error>
```

- **mid** Mail ID in which the error occurred.
- **code** Error codes. We made error list at Section 3.5.
- **target** Send back the transaction XML s/he received at last.

Request to Restart Session

```
<restart>
  <mid>83f93ac93dabd342</mid>
  <sid>43eaf9bd220a1c88</sid>
  <expire>20011103204732</expire>
</restart>
```

- **mid** Unique ID provided to the Certified Mail. When a party want to restart, MID is carried over to new secret exchange session.
- **sid** When the secret exchange session restarts, each party generate SID again correspond to new session.

- **expire** Expiration time. This time corresponds to SID. Expiration time is generated again along with regenerating the SID.

Reply to Restart Request

```
<restart_rep>
  <mid>83f93ac93dabd342</mid>
  <sid>a0fe8dbb8ea334dc</sid>
  <expire>20011103204812</expire>
</restart_rep>
```

- **mid** Unique ID provided to the Certified Mail. When a party want to restart, MID is carried over to new secret exchange session.
- **sid** When the secret exchange session restarts, each party generate SID again correspond to new session.
- **expire** Expiration time. This time corresponds to SID. Expiration time is generated again along with regenerating the SID.
- **mid** Appoint the MID to terminate.

Chapter 5 Consideration and Improvement

5.1 Consideration

Now, we consider how this system resolves the problems we assumed at the previous section. Recent that, we made the following assumptions.

1. Alice sent a message and Bob received it. But Bob claims that he did not receive it.
2. Alice did not send a message to Bob. But she claims that she has certainly sent it to Bob.
3. Alice sent a fake message to Bob, and Bob received it. But Alice claims that she sent the genuine one to Bob.

For each problem, users can deal with them as follows with this system.

1. Alice can protest by showing Bob's receipt, which includes the commitment. The receipt all together includes the transformed secret key and parameters Alice sent. So Alice can claim that the receipt perfectly corresponds to the message she sent. In case Bob had sent an invalid receipt, Alice should show whole the communication log, and she can represent that the contents of the receipt differ from the abstract he received.
2. As long as both parties agree to use this system, Bob requires Alice to show his receipt. If she did not send the mail, she cannot get the decrypted receipt.
3. In this case, Bob can allege the difference between abstract and the message he received by showing the first mail and following secret exchange.

Chapter 6 Reducing the Number of Sessions

Although we selected Okamoto and Ohta's protocol for their efficiency, over 100 transactions are still needed to exchange 64 bits of secret keys. Thus we want to reduce the number of sessions to reduce the time needed to complete procedures. Considering how to deal with this problem, we compared the following three approaches.

1. Reduce the length of key bits.
2. After some bits have been sent, send the whole bits leftover at once.
3. After some bits have been sent, double the number of bits sent at once in each round.

The first method is very simple, but has a problem in the view point of the cryptographic strength apparently. The strength of a key is basically exponential in the key length. So we cannot adopt this approach to reduce the number of transactions.

The second method is safer than the first, but this method ignores the fairness discussed in Section 2.2.4. We need to care about the case in which one party has sent whole the remaining bits, but the other receives it without sending the remaining ones. In this case the fairness between two parties are broken apparently. So we cannot adopt this approach, either.

In the last method, they double the number of bits they send at the next round every time. the following is an example: Alice and Bob in turn send each other the first 32 bits of their 64 bits secret keys. They send 1 bit at the 33rd step, 1 bit at the 34th step, 2 bits at the 35th step, 4 bits at the 36th step, and so on. Each party can send the last 32 bits within 6 steps. More precisely, each party sends some amount of bits at once at the first step, where this amount depends on th cryptographic strength. After that, each party sends bits by doubling the number. As an extreme case, if we start doubling from the first step, each party sends 2^{i-1} bits at the i th step. In this case, n bits are exchanged within $O(\log n)$ steps.

Suppose that at some moment during the procedure, Alice and Bob have received some amount of bits, and they need to consume t_A and t_B of time, respectively, to calculate the true message by, for example, an exhaustive search. Although the fairness is originally defined as the maximum *ratio* of t_A and t_B and is always constant in

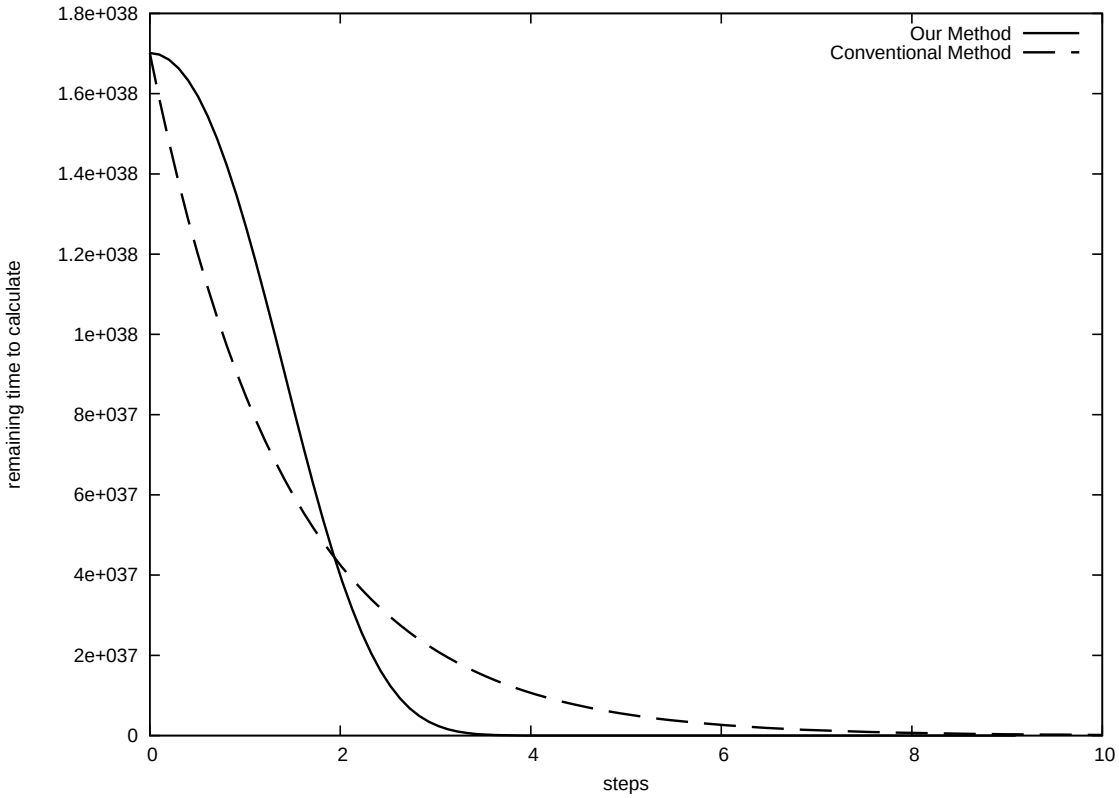


Figure 4: The decreasing of differences

Okamoto and Ohta's protocol, it is also natural to define it as the maximum *difference* between them. It is not hard to see that in our last method, this quantity is maximum at the first step and decreases as the protocol proceeds as we show in Fig. 4, namely, the fairness is the same as the case of bit-by-bit protocol from the viewpoint of the worst case analysis. Nevertheless, we have an advantage of reducing the number of rounds exponentially. Hence, we decide to use the last method in our designs described later.

Chapter 7 Conclusion

We designed a certified e-mail exchange system based on simultaneous secret exchange protocol under the following assumptions.

- Both Alice (the sender) and Bob (the receiver) agree to exchange her message and his receipt. They only want to avoid unfair affairs.
- E-mail is the only method Alice and Bob can exchange messages.

The former is needed to limit the problems, as 'selective receipt' cannot be prevented fundamentally without TTPs. We made the latter assumption to guarantee the reachability considering the real environments with many barriers e.g. firewalls.

The problems that our system should resolve are listed as follows.

1. Alice sent a message and Bob received it. But Bob claims that he did not receive it.
2. Alice did not send a message to Bob. But she claims that she has certainly sent it to Bob.
3. Alice sent a fake message to Bob, and Bob received it. But Alice claims that she sent the genuine one to Bob.

At first, we design the entire procedure mainly based on Okamoto and Ohta's simultaneous secret exchange protocols, adding processes to confirm the contents each party have received in Section 3.1. We selected their protocols because of its efficiency, as we discussed in Section 2.2.5.

In this system, we want to realize good properties often referred in certified mail researches with TTPs, for example 'send-and-forget'. So we assume two players, an MTA (Mail Transfer Agent, e.g. SMTP Server) and an MUA (Mail User Agent e.g. the client the sender or the receiver use) in each party. This approach enabled the system to realize 'send-and-forget' for the sender when the sender can control an MTA, or exchange messages for the sender with the receiver directly otherwise. We discussed this problem in Section 3.2.

Considering from the assumptions, we adopt e-mails to the method of exchanging transactions. So we designed each transaction formats on XML and attach it to e-mails in Section 4.1.

We then considered how this system works in Section 5.1. Although we confirmed that this system actually solves the problems we assumed, it seems to take rather long time in some situations. This is because there are over 100 transactions between Alice and Bob until they complete the entire sessions, so the delay of e-mails are greatly amplified.

To resolve this problem, we propose a key sending method in Section Chapter 6. The ‘fairness’ is defined as the maximum of the difference of time needed to calculate the remaining bits of keys at any time, where the difference is taken between two players. This amount is ‘at most twice’ in a normal simultaneous secret exchange protocols, while it is ‘at most constant difference’ in our method.

Acknowledgments

I would like to give my sincere thankfulness to my supervisor, Prof. Yasuo Okabe. He gave me correct guidance, showed me a appropriate direction, and encouraged. Because of his guidance, I was able to work this research from a simple interest. I am extremely grateful to Assoc Prof. Shuichi Miyazaki for empathic guiding, suggestion and coaching thesis. I also thanks to Assoc Prof. Motonori Nakamura for good advices. I would also like to thank Ms. Kazumi Sakai and office staff members for proper clerical jobs.

I wish to express my sincere gratitude to all the members of Okabe Laboratory. Especially My peer Mr. Keiji Maekawa taught me many useful and interesting topics and being in friendly rivalry with me. Mr. Shin Maruyama gave me an opportunity of training and social experience. Mr. Koji Kobayashi, Mr. Naoyuki Morimoto greatly helps my questions and researches. And I am do thank the all other members in this laboratory.

I also want to thank to Prof. Okamoto and Prof. Ohta. This research work is greatly based on their researches and I was helped much by their books.

Finally I want to be thankful to my mother and my family about having always supported me, and at last my father in heaven, who encouraged me always.

References

[1] Fajman, R.: An extensible message format for message disposition notifications, RFC 2298 (Proposed Standard) (1998). Obsoleted by RFC 3798.

[2] ISO/IEC: ISO 13888-1 IT security techniques Non-repudiation Part 1: General (1997).

[3] Zhou, J. and Gollmann, D.: Observations on non-repudiation, *Proc. ASIACRYPT*, pp. 133–144 (1996).

[4] Asokan, N., Schunter, M. and Waidner, M.: Optimistic protocols for fair exchange, *Proceedings of the 4th ACM conference on Computer and communications security*, ACM New York, NY, USA, pp. 7–17 (1997).

[5] Imamoto, K. and Sakurai, K.: A scalable on-line certified E-mail protocol using password authentication, *WISA2002*, pp. 319–331 (2002).

[6] Ben-Or, M., Goldreich, O., Micali, S. and Rivest, R. L.: A fair protocol for signing contracts, *IEEE Transactions on Information Theory*, Vol. 36, No. 1, pp. 40–46 (1990).

[7] Rabin, M.: How to exchange secrets by oblivious transfer, Technical report, Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981 (1981).

[8] Luby, M., Micali, S. and Rackoff, C.: How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin, *Proc. FOCS*, pp. 11–21 (1983).

[9] Blum, M.: How to exchange (secret) keys, *ACM Trans. Comput. Syst.*, Vol. 1, No. 2, pp. 175–193 (1983).

[10] Cleve, R.: Controlled gradual disclosure schemes for random bits and their applications, *Proc. CRYPTO*, pp. 573–588 (1989).

[11] Even, Shimon, O. G. and Lempel, A.: A randomized protocol for signing contracts, *Communications of the ACM*, Vol. 28, No. 6, pp. 637–647 (1985).

[12] Okamoto, T. and Ohta, K.: How to simultaneously exchange secrets by general assumptions, *Proceedings of the 2nd ACM Conference on Computer and communications security*, pp. 184–192 (1994).

[13] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1, RFC 2616 (Draft

Standard) (1999). Updated by RFC 2817.

[14] Wood, D.: *Programming Internet Email*, O'REILLY (1999).

[15] SIProp Project/Noritsuna Imamura, Hirotaka Nisato, T. S. M. E.: *Private Implementation Protocol*, Mainichi Communications (2007).