Master Thesis

Universally Composable Mix-net, Blind Signatures and Voting

Supervisor Professor Tatsuaki Okamoto

Department of Social Informatics
Graduate School of Informatics
Kyoto University

SEIJI DOI

February 15, 2006

Universally Composable Mix-net, Blind Signatures and Voting

SEIJI DOI

Abstract

An electronic voting system is a technology that is expected to be available in the future. It will decrease the cost of voting; we will be able to vote anytime and anywhere by using PC from our home, or a cellar phone. Voting protocols must have two security properties, *Receipt-freeness* and *Universal verifiability*. *receipt-freeness* means that anyone cannot prove the content of his vote to someone else. *universal verifiability* means that anyone can verify the result of voting. Many techniques have been developed to realize the system. Mix-net and blind signatures are the most popular techniques. Mix-net is a set of mix servers. Given a list of messages, a server randomizes the sort of the list (*shuffles*) and outputs the resulting list to the next server. By repeating this task the final output messages realize anonymous. Blind signatures also realize anonymity. In blind signatures, when a signer signs a document, the signer cannot see the content of the document. In the case of voting systems, a special party such as administrator or voting committee generates a blind signature for each ballot to prevent multiple voting and voting from invalid parties.

Canetti recently introduced the universal composability (UC) framework as a new approach for analyzing the security of cryptographic primitives and protocols. In the UC framework, it is guaranteed that a secure primitive/protocol maintains its security even if other primitives/protocols run concurrently. It is called *universally composable theorem*. It allows us to use a protocol like as a module, therefore, we can combine protocols and construct large systems easily. In the UC framework, we consider two worlds; real world where an actual protocol is executed, and ideal world where there exists an ideal functionality instead of the protocol. If a distinguisher cannot distinguish the two worlds by interaction, it seems that the protocol is essentially equal to the functionality and is said to be secure in the UC framework (UC-secure).

Since the UC security requirement is very strong, it raises the new question of whether the conventional security notions satisfy the UC security. It is a hot

topic to compare the UC security with the conventional security. For example, a UC-secure signature scheme is equivalent to a secure (*existential unforgeable against chosen-message attacks*) signature scheme.

In this paper, we discuss voting systems in the UC framework. A voting system consists of some cryptographic primitives such as key authentication, encryption/decryption, thus universally composable theorem can be applied to voting systems. To evaluate voting protocols in the UC framework, we first define a functionality of voting, $\mathcal{F}_{\mathrm{VOTE}}$.

Next, we introduce a protocol that securely realizes $\mathcal{F}_{\mathrm{VOTE}}$ in hybrid model with two different functionalities; a functionality of Mix-net with sender authentication and a functionality of tallying ballots. This construction is simple and easy to understand by intuition. We also define these two functionalities. The hybrid protocol guarantees receipt-freeness and universal verifiability. Additionally, we introduce how to securely realize the two functionalities.

Next, we evaluate blind signatures for another approach to construct a voting system in the UC framework. We then compare the conventional security with the UC security of blind signatures. Security definition of blind signatures is formulated by Juels, Lindell and Ostrovsky. Blind signatures must have two security properties, *blindness* and *non-forgeability*. Blindness is the security when a signer is corrupted and means that any party cannot linkage a signature and the information during the interaction with a signer to generate the signature. Non-forgeability, defined by Pointcheval and Stern as *one-more forgeability*, is the security when a user is corrupted and means that any user who wants to obtain signatures cannot get more signatures than the number of interactions with a signer. In this research, we define stronger security than that of Juels et al. The stronger security is appropriate in many applications.

In conclusion, we define a functionality of voting in the UC framework, and introduce a hybrid protocol that securely realizes the functionality, with two different functionalities. We also introduce how securely realize these two functionalities. Additionally, we define a functionality of blind signatures. We compare the conventional securities with the UC security.

汎用的結合可能な Mix-net, ブラインド署名および電子投票

土居誠司

内容梗概

将来その実現が望まれる暗号プロトコルの 1 つに電子投票がある。ここで述べる電子投票とは自宅のパソコンや携帯電話等の機器からネットワークを通じて行うことのできる投票を指す。電子投票プロトコルが満たすべき性質には「無証拠性」と「全体検証性」の 2 つがある。無証拠性とは、いかなる投票者も自身の投票内容を証明することができない性質であり、全体検証性とは、投票が正しく行われたことをプロトコルの全ての参加者が検証できる性質である。これまでいくつかの投票プロトコルが提案されているが、 Mix-net を利用したもの、 blind 署名を利用したものがその代表例である。

一方、暗号プロトコルの安全性を統一的に評価するための枠組みとして、近年 Canetti によって「Universally Composable (UC) framework」が提案された。この framework では「universally composable theorem (UC 定理)」が成り立つ。この定理により UC framework で安全なプロトコルは他のプロトコルと組み合わせて用いられてもその安全性が保証される。 UC framework が提案される前までは、提案プロトコル毎にそれぞれ独立して安全性を評価しており、たとえそのプロトコルが単体で安全であったとしても他のプロトコルと組み合わせた場合の安全性を保証するものではなく、結合プロトコル全体を改めて評価する必要があった。UC 定理によってプロトコルをハードウェアモジュールのように扱うことができ、大規模な暗号プロトコルの構築を小さなプロトコルを組み合わせて容易にでき、その安全性の評価も容易に行える。 Canetti が UC framework を提案して以降、公開鍵暗号やゼロ知識証明等既存のプロトコルの安全性と UC での安全性の比較が盛んに行われている。 UC framework では、実際のプロトコルが動作する現実世界と、そのプロトコルの理想機能が動作する理想機能が動作する理想世界との 2 つの世界を考える。この 2 つの世界とある判別者が interaction を行い、区別がつかなければそのプロトコルは理想機能を安全に実現していると言い、 UC 安全と言う。

本研究では、 UC framework における電子投票プロトコルについて考察する。電子投票プロトコルは鍵認証、票の暗号化等多くの暗号機能から成るので、 UC 定理の有効性が高い。 Mix-net はメッセージに匿名性を与えるプロトコルであり、

mix-server の集合からなる。各 server が入力として受けたメッセージをシャッフルし次の server へ渡す。この操作を繰り返すことで最終的に出力されるメッセージは匿名になる。 blind 署名は署名者が署名する文書を見ることなく署名をする技術である。投票においては、選挙管理機関が投票に blind 署名を与えることで、投票の匿名性を保つと同時に正当な投票者を識別できる。匿名性の実現は投票プロトコルの無証拠性の実現に必要である。全体検証性は、 mix-net においては各 server の動作の正しさを検証することで実現され、 blind 署名では、公開鍵による署名の検証で実現される。

UC framework において投票プロトコルを評価するために、まず投票プロトコルの理想機能を定義する。続いて Mix-net を用いた実現方法について考察する。投票者認証機能を持つ Mix-net の理想機能と、票の集計を行う理想機能を定義し、この 2 つの理想機能と動作する hybrid モデルで、投票の理想機能を実現するプロトコルを提案する。併せて、この hybrid プロトコルに用いられている 2 つの理想機能の実現法を示す。

次に、投票プロトコルのもう 1 つの実現手法である blind 署名について考察する。 Mix-net の時と同様に、 blind 署名を UC framework で投票プロトコルに応用するために、 blind 署名の理想機能を定義する。そしてこれまでに提案されてきた blind 署名プロトコルの安全性と UC での安全性との比較を行う。その際、これまでの安全性よりも強い安全性を定義し、その強い安全性よりも UC での安全性のほうが強いことを示す。

結論として、今回の研究では、暗号プロトコルの安全性の統一的な枠組みである UC framework において、電子投票の理想機能を定義し、それを実現するプロトコルを 2 つの理想機能を用いた hybrid モデルで提案した。続いて 2 つの理想機能を実現する方法を提案した。また、投票プロトコルを実現するための代表的な技術である blind 署名を UC framework で評価するために、 blind 署名の理想機能を定義し、これまでに提案された blind 署名の安全性と UC framework における安全を比較した。その際、これまでの blind 署名の安全性よりも強い安全性を定義し、その強い安全性よりも UC framework での安全性のほうが強いことを示した。

Universally Composable Mix-net, Blind Signatures and Voting

Contents

Chapter 1 Introduction

An electronic voting system is a technology that is expected to be available in the future. It will decrease the cost of voting; we will be able to vote anytime and anywhere and we can obtain the result immediately without spending much effort. Many techniques have been developed to realize voting systems. The most popular techniques are;

- Mix-net
- Blind signatures
- Homomorphic functions

Homomorphic functions only works for one bit voting, where voters can votes 'yes' or 'no'. Thus its application is restricted. We now suppose voting systems that allows voters to write a character string of some length. In this research, we discuss Mix-net and blind signatures that allow voters to vote a messages of some length as their ballots. A Mix-net is a set of mix servers. Given a list of messages, a server randomizes the sort of the list (*shuffles*) and outputs the resulting list to the next server. By repeating this task messages get anonymity. Blind signatures are an extension of digital signatures. By using this technique, a signer generates a signature without seeing the document to be signed. In the case of voting, a voting committee generates blind signatures only to ballots from eligible voters. We obtain the result by counting only ballots that passes the verification.

Canetti introduced the universal composability (UC) framework as a new approach for analyzing the security of cryptographic primitives and protocols. Cryptographic protocols are used in various situations such as networks, e-commerce, and data management. Cryptographic protocols become more important. On the other hand, they have recently gotten bigger and complicated. It also is complicated to analyze their security. In the case that a protocol consists of some sub-protocols, the composed protocol is not always secure even if each sub-protocol is secure individually. We must analyze the composed protocol anew. It requires great efforts.

In the UC framework, it is guaranteed that a secure primitive/protocol maintains its security even if other primitives/protocols run concurrently. It is called *universally composable theorem*. It allows us to use a protocol like as a module, therefore, we can combine protocols and construct large systems easily. It decrease the efforts for analysis of the security of a large composed protocol.

In the UC framework, we consider two worlds; real world, where an actual protocol, and ideal world where there exists ideal functionality instead of the protocol. If a distinguisher cannot distinguish the two worlds by interaction the protocol is essentially equal to the functionality and is said to be secure in the UC framework (UC-secure).

Since the UC security requirement is very strong, it raises the new question of whether conventional security notions satisfy UC security. It is a hot topic to compare the UC security with conventional security. For example, a UC-secure signature scheme is equivalent to a secure (*existential unforgeable against chosen-message attacks*) signature scheme. We thus consider about the UC security of voting protocols. Additionally, we consider about the UC security of Mix-net and blind signatures.

1.1 Our results

In this research, we discuss about a voting system in the UC framework. We take up mix-net and blind signatures as techniques that realize voting systems. First, We formulate the security of voting systems in the UC framework (i.e., define the ideal functionality of voting systems), and show a voting protocol that is secure in the UC framework using mix-net technique. The protocol realizes the functionality of voting under the reasonable assumption.

Additionally, we show that the conventional security of blind signatures is truly weaker than the UC security. That is, first we formulate the security of blind signatures in the UC framework and show that the class of UC-secure blind signatures is a proper subset of that of secure (in the sense of [14]) blind signatures, assuming a one-way trapdoor permutation.

We then introduce a stronger security definition (stronger blindness; SB-security) of blind signatures than that by Juels et al. [14]. SB-security is more

appropriate in many applications (e.g., electronic cash and voting) than Juels et al.'s. We then show that SB-security of blind signatures is also truly weaker than security in the UC framework. That is, we show that the class of UC-secure blind signatures is a proper subset of that of SB-secure blind signatures, assuming a one-way trapdoor permutation.

Chapter 2 Preliminaries

2.1 Voting

Voting protocols are required to be available in the future. Current 'paper-based' voting systems spends too much cost and manpower. If electronic voting systems are realized, we can greatly decrease them. Additionally, we may realize direct elections of large scale that are not realistic now. Generally, voting systems will be important infrastructure so that will require very high level security. Invalid voting, buying and selling votes, artificial manipulation of the result and getting the progress of the voting are given as considerable attacks to voting systems.

Voting protocols work voters and some other parties. Voters vote their ballots and they can obtain the final result later. Voting protocols must have two security properties, *receipt-freeness* and *universal verifiability*. *Receipt-freeness*, introduced by Benaloh and Tuinstra [3], means that anyone cannot prove the content of his vote to someone else. It is a critical property. If a voter can prove the content of his vote, he can sell his vote. Consider the following example. Alice and Bob come forward as candidates in an election, and Alice intends to buy voters. If receipt-freeness is not guaranteed then a voter may obtain some money due to show the proof that he votes Alice. In the current voting system using paper, *receipt-freeness* is also guaranteed. *Universal verifiability* means that anyone can verify the result of voting. When some dishonest acts are carried out, anyone notice it by the verification. In the current voting system, it is guaranteed by believing that tally is done honestly. Even if some manipulation is added to a tally result, we cannot notice it.

Benaloh and Tuinstra firstly showed a receipt-free voting scheme. It, however, requires the usage of a *voting booth* in which only designated party can communicate with another party, and the communication is perfectly secret for all other parties. The existence of voting booth is very strong assumption. Sako and Killian showed a receipt-free voting scheme using mix-type protocol assuming the existence of an *untappable channel* [20]. An untappable channel is more practical than a voting booth but strong assumption. By an untap-

pable channel only designated voter can send a message to another party, and the message is perfectly secret to all other parties. Okamoto shows a receipt-free voting protocol using a blind signature scheme [18]. The protocol is more practical than former two protocols and requires untappable channel or voting booth. Like these schemes, voting systems often need strong assumptions. It has been target that how to decrease these assumption with maintaining the security.

2.2 Mix-net

A mix-net is a technique that makes messages anonymous. The concept of mix-net is introduced by Chaum [9]. The image of the behavior of a mix-net is shown in Figure 2.1. The idea for making anonymous is intuitive. A mix-net is a family of mix servers. Each server takes as input a list of messages and then it shuffles messages by randomizing the order of the list. It sends as output the randomized list to the next server. Repeating this computation, we can make messages be anonymous, which guarantees receipt-freeness in a voting protocol. If at least one server works honestly then the anonymity is maintained. After the computation, each server proves that it has worked honestly according to the protocol using zero knowledge proof. Anyone verify these proofs, which means that universal verifiability is guaranteed.

As above, each mix-server must prove that it honestly proceeds according the protocol in zero-knowledge manner. It takes much complexity. Thus, efficient protocols have been required. Many literatures about mix-net techniques have been written up to now. Abe gives an efficient construction of a general mix-net, and argues about its properties [1]. Jakobsson has written (partly with Juels) more general papers on the topic of mixing focusing on efficiency. [13].

2.3 Blind signatures

The concept of blind signatures was first introduced by Chaum [9]. Blind signatures are extended variants of digital signatures. Digital signatures are used for binding between documents and party such as person or organization. Digital signatures are close to our life and embedded in many applications such as

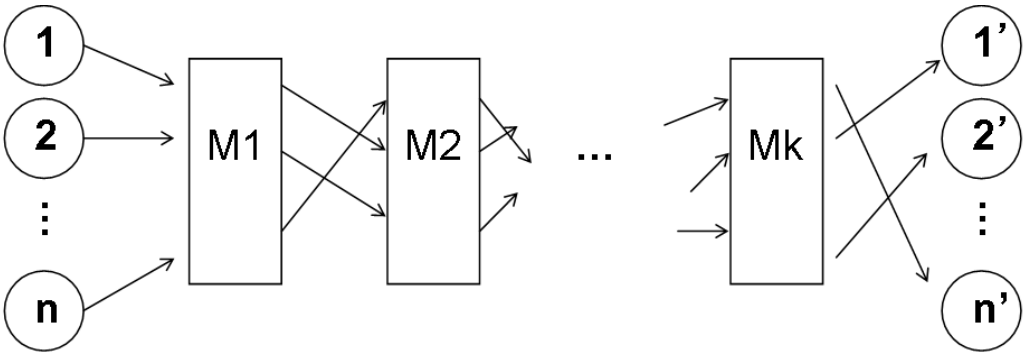

Figure 2.1: A Mix-net

e-mail, e-commerce, or online shopping. In blind signatures, a user interacts with a signer for signature generation, and the signer cannot see the content of the document that he is signing. Before generating a signature, a user randomizes a document to conceal it. The user sends the randomized document to the signer. The signer generates a signature to the randomized document by using his secret key and then returns the signature to the user. The user transform the received signature into a signature that passes the verification with the original (not randomized) document. The user outputs the resulting signature as a signature of the original document. Anyone can verify the signature by using the signer's public key. Blind signatures are significant cryptographic techniques and are used to construct anonymous electronic cash and electronic voting systems.

Some variants of blind signatures have been developed. Chaum introduced a RSA blind signature scheme. The security proof of the scheme is based on the discrete logarithm problem. Okamoto introduce variants based on Fiat-Shamir signature scheme and Schnorr signature scheme [17].

Pointcheval and Stern introduced a security notion called *one-more forgery* [19] to define the security of blind signatures. In a blind signature scheme that is secure against one-more forgery, a user cannot get more signatures than the number of interactions with the signer [19]. Juels, Luby and Ostrovsky introduced a general definition of blind signatures and formulated two security properties: *blindness* and *non-forgeability* [14]. Blindness means that the adversary cannot link a signature to the corresponding message and non-forgeability

means that the adversary cannot get more signatures than the number of interactions with the signer even if the adversary undertakes, with the signer, adaptive, parallel and arbitrarily interleaved interactive protocols. They called such an attack *an adaptive interleaved chosen-message attack*. They also showed that there exists a blind signature protocol that is secure (i.e. satisfies the above two security properties) against an adaptive interleaved chosen-message attack if one-way trapdoor permutations exist.

Chapter 3 Universal Composability

The Universal Composability security framework was proposed by Canetti [4]. The main concern was to create a new approach to the assessment of cryptographic protocols. In traditional approaches, protocols were defined for isolated execution. This is sufficient for small or simple protocols and evaluating their security is relatively easy. However, recent most of cryptographic systems consist of many protocols and thus these protocols must run in a complex environment where the protocols may run simultaneously or concurrently. In such a situation, analyzing protocols by the traditional approach is a virtually impossible. Even if every component protocol is secure by itself, the security of the compound protocol must be analyzed anew.

In the UC framework, a protocol is defined as stand-alone but it is guaranteed to run security in any arbitrary environment. The framework guarantees that a compound protocol consisting of just secure protocols is also secure, which is called *the universal composition theorem*. Thus, when analyzing a multi-protocol system, we analyze each protocol in isolation, and the protocol maintains its security when it is running within an environment wherein another protocol may be running concurrently. The universal composition theorem ensures that a cryptographic system consisting of secure protocols is secure. We show an instance of universally composable theorem in Figure 3.1.

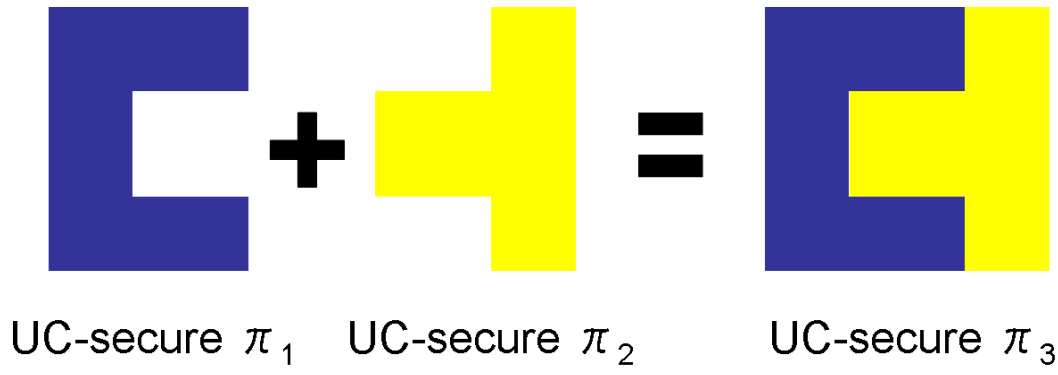

Figure 3.1: Universally Composition theorem

The advantage of using this framework is that protocols can be handled as

modules. As mentioned above, we can easily construct large secure systems by combining small and simple secure protocols. However, the security of protocols must be proved by the unique and technical method of simulation-based proof, and it takes more effort than standard complexity-based proofs.

We describe here a method of proving the security of a protocol in the UC framework. In the framework there are two worlds: the ideal world and the real world. In the real world, there are parties and the adversary. Parties interact with each other according to the protocol. The adversary can control the delivery of the messages that the parties send and may corrupt some parties during the execution. When a party is corrupted, it has to follow the instructions of the adversary. In the ideal world, there are parties, an ideal functionality, and the simulator. An ideal functionality is the ideal behavior of the function that a protocol should achieve. Parties cannot directly interact with each other. They exchange messages and interact via the functionality. The simulator can control message delivery and can corrupt parties as well as the adversary.

Furthermore we also consider the middle world, called the hybrid world, which lies between the ideal world and the real world. In the hybrid world, parties interact according to a protocol as in the real world. Additionally there are functionalities that can be accessed by all parties. There also exists the adversary, where for the interaction of a party and the functionality it plays the role of the simulator and for the interaction among parties it plays the role of the adversary.

The UC framework also sets the special party called the environment, which interacts with each world. It can give as input to the parties arbitrary strings and can order the adversary to corrupt parties. It tries to distinguish these two worlds by the interactions. After an interaction, the environment outputs a bit. The environment distinguishes the two worlds if the difference between the probability that the environment outputs 1 after the interaction with the real world and that after the interaction with the ideal world is non-negligible. We say a protocol is *UC-secure* if for any adversary there exists a simulator such that for any environment the ideal world and the real world are indistinguishable. Moreover, we say such a protocol *UC-realizes* the functionality in the ideal

world.

Here, we show how an environment distinguishes the two worlds. For example, a protocol running in the real world may generate some internal data that is not created in the ideal world. Therefore, the environment can let the adversary corrupt a party and it uses the internal data to distinguish the two worlds. To prevent this, the simulator in the ideal world should generate all necessary data and try to make the interaction with the environment look like the one in the real world. The simulator runs parties and the adversary in itself and simulates the interaction in the real world. In the simulation it also behaves as the environment and sends parties input convenient for the simulation. It gets back various data from the simulation and sends the data to the environment. We present an image of an interaction between an environment and the two worlds in Figure 3.2.

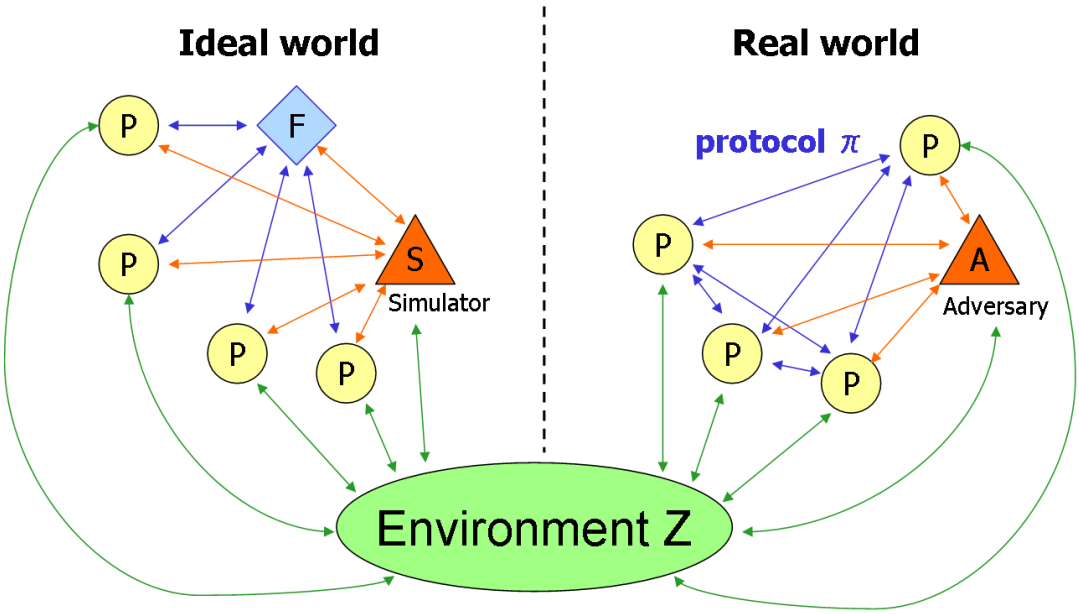

Figure 3.2: An interaction of the environment and two worlds

3.1 Adversarial model

In the UC framework, two types of corruption by adversaries, *static* and *adaptive*, are considered. In the case of static adversaries, the identities of the

corrupted parties are fixed before the computation starts. On the other hand, in the case of adversaries, adversaries may corrupt parties as the computation proceeds. Although the security against static adversaries is weaker than the security against the security against adaptive adversaries, there exist some values to analyze the security against static adversaries as follows,

- The definition and the proof are considerably simpler.
- Some cryptographic protocols have security proofs that hold only against static adversaries.

The security of adaptive adversaries are very strong. Protocols that are secure against adaptive adversaries must maintain their security even if parties works honestly during a computation and at the end of the computation an adversary corrupt parties and get internal data. It seems impossible for protocols in which parties use secret data such as secret key and parameters of randomization.

Chapter 4 Ideal functionalities for voting

In this chapter, for analyzing the security of voting protocols in the UC framework, we define a functionality of voting. Additionally, we define two functionalities that is used to realize the voting functionality. The realization of the voting functionality is showed in Chapter 5.

4.1 A functionality of voting

In this chapter, we introduce a functionality of voting, $\mathcal{F}_{\text{VOTE}}$ and evaluate the security of a voting protocol in the UC framework. We present the definition of $\mathcal{F}_{\text{VOTE}}$ in Figure 4.1. Messages delivered between the functionality and parties usually includes a session ID, *sid* [4]. We hereafter omit the description of *sid* for simplification.

$\mathcal{F}_{\text{VOTE}}$ has an interface with voters $P_1, \cdots, P_n$, with vote servers $M_1, \cdots, M_k$ and with simulator $\mathcal{S}$. The main role of $\mathcal{F}_{\text{VOTE}}$ is counting ballots and publishing the result. $\mathcal{F}_{\text{VOTE}}$ has a list of eligible voters and only counts ballots from the eligible voters. It has a list of candidates and outputs the number of ballots for each candidates. The contents of invalid ballots are not output but only the number of them is output.

As shown Figure 4.2, we UC-realize $\mathcal{F}_{\text{VOTE}}$ by two step. In the first step, we define two functionalities $\mathcal{F}_{\text{AMN}}$ and $\mathcal{F}_{\text{TALLY}}$, and then show a hybrid protocol π_{VOTE} that realizes $\mathcal{F}_{\text{VOTE}}$ with the two functionalities. The two functionalities are defined in 4.2 and 4.3. In the second step, we introduce how to UC-realize the two functionalities.

4.2 An ideal functionality of a mix-net with authentication

In this chapter, We define a functionality of a mix-net with authentication, $\mathcal{F}_{\text{AMN}}$, that shuffles messages from authenticated senders. We present the definition of $\mathcal{F}_{\text{AMN}}$ in Figure 4.3. $\mathcal{F}_{\text{AMN}}$ has the list of senders. When $\mathcal{F}_{\text{AMN}}$ receives messages from senders it sees the list and adapts shuffling only to the messages from the senders in the list. $\mathcal{F}_{\text{AMN}}$ receives only one message from one sender.

$\mathcal{F}_{\text{VOTE}}$

With voter $P_1, \cdots, P_n$, vote server $M_1, \cdots, M_k$ and simulator $\mathcal{S}$. $\mathcal{F}_{\text{VOTE}}$ has a list of candidates $L_C = \{C_1, \cdots, C_{N_C}\}$ and a list of eligible voters $L_P = \{P_1, \cdots, P_{N_P}\}$.

1. $J_M = \phi, c_i = 0 (i = 1, \cdots, N_C), c_{inv} = 0$.
2. Upon receiving $(\texttt{Vote}, v_i)$ from P_i, if $P_i \in L_P$ then send $(\texttt{Vote}, P_i)$ to $\mathcal{S}$, and then,
   - if $v_i = C_j (C_j \in L_C)$ then $c_j = c_j + 1$.
   - otherwise $c_{inv} = c_{inv} + 1$.
   - ignore future message from P_i such as formed $(\texttt{Vote}, \cdot)$.
3. Upon receiving $(\texttt{Start})$ from M_j, $J_M \leftarrow J_M \cup \{j\}$ and send $(\texttt{Start}, M_j)$ to $\mathcal{S}$. Furthermore, if $|J_M| \geq \frac{k}{2}$ then compute $L' = \{\{c_j\}_{j=1}^{N_C}, c_{inv}\}$.
4. Upon receiving $(\texttt{Read}-\texttt{result})$ from P_d, send $(\texttt{Read}, P_d)$ to $\mathcal{S}$.
   - If L' is computed then return $(\texttt{Result}, L')$ and send $(\texttt{Result}, L')$ to $\mathcal{S}$.

Figure 4.1: Functionality $\mathcal{F}_{\text{VOTE}}$

Wikström defined a functionality of Mix-net, $\mathcal{F}_{\text{MN}}$, and introduced a protocol that UC-realizes $\mathcal{F}_{\text{MN}}$ [22]. We extend $\mathcal{F}_{\text{MN}}$ and define $\mathcal{F}_{\text{AMN}}$.

4.3 An ideal functionality of tally

In this chapter, we define a functionality of tally, $\mathcal{F}_{\text{TALLY}}$. The main role of $\mathcal{F}_{\text{TALLY}}$ is encryption and count of ballots.

The reason that $\mathcal{F}_{\text{TALLY}}$ encrypts ballots to be voted is as follows. Assume that, in a voting protocol that consists of mix part and tally part, voters votes ballots as plaintext, then a malicious tally server can read the contents of ballots when it counts the ballots.

We present the behavior of $\mathcal{F}_{\text{TALLY}}$ in Figure 4.4. $\mathcal{F}_{\text{TALLY}}$ works with observers $P_1, \cdots, P_n$ and tally servers $T_1, \cdots, m, T_k$ and a simulator $\mathcal{S}$. We assume that a list of candidates L_C is published First, when $\mathcal{F}_{\text{TALLY}}$ receives a request to generate keys for encryption and decryption, it ask S to generate keys. When $\mathcal{F}_{\text{TALLY}}$ receives a request for encryption with a message and public key from P_i,

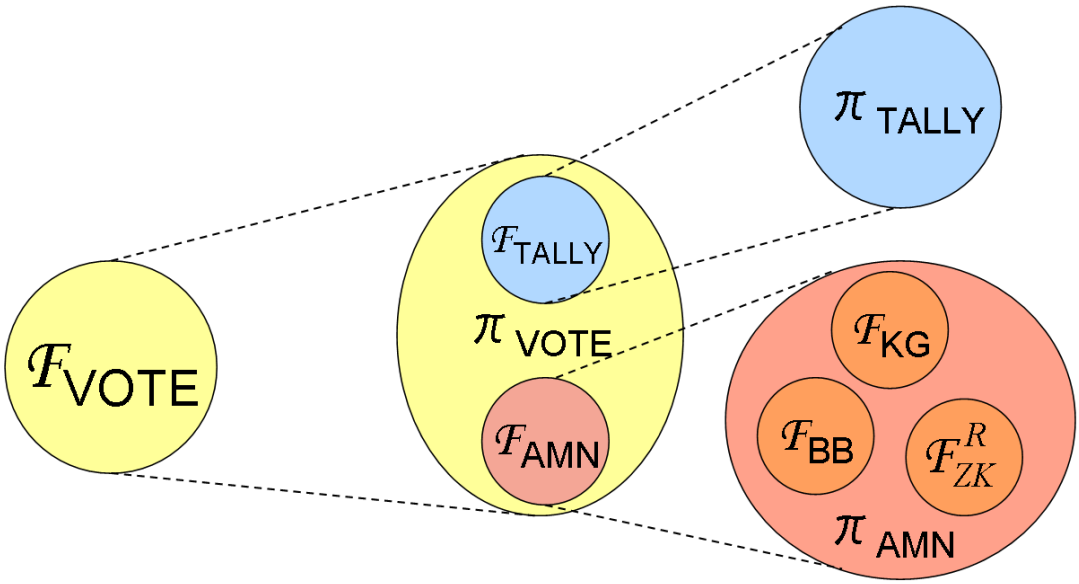

Figure 4.2: The overview of the construction

$\mathcal{F}_{\text{AMN}}$

With sender $P_1, \cdots, P_n$, mix-server $M_1, \cdots, M_k$ and simulator $\mathcal{S}$. $\mathcal{F}_{\text{AMN}}$ has a list L_P

1. $L = \phi, J_P = \phi, J_M = \phi$
2. Upon receiving $(\texttt{Send}, m_i)$ from P_i,
   - if $P_i \in L_P$ then add i to J_P, add m_i to L and $(\texttt{Send}, P_i)$ to $\mathcal{S}$.
   - otherwise ignore.
3. Upon receiving $(\texttt{Run})$ from M_j, add j to J_M,
   - if $|J_M| \geq \frac{k}{2}$ then sort L lexicographically to form L', send $(\texttt{List}, L', M_j)$ to $\mathcal{S}$ and send $(\texttt{List}, L')$ to all mix-servers.
   - otherwise send $(\texttt{Run}, M_j)$ to $\mathcal{S}$.

Figure 4.3: Functionality $\mathcal{F}_{\text{AMN}}$

if the public key is already published then they encrypts the message with the public key, and returns the ciphertext. Otherwise, it encrypts a fixed message that may have no meaning, and returns the ciphertext. When $\mathcal{F}_{\text{TALLY}}$ receives a message of tally with a list from P_i, if it receives $(\texttt{Run})$ from more than half of tally servers send $(\texttt{Run})$ then it counts the elements of the list according to L_C. When it receives a message to publish the result, if the count is completed then

return the result. The result forms as a list and the list include the number of ballots for each candidate and the number of invalid ballots. The contents of invalid ballots are published so that anyone cannot learn what is written on invalid ballots, which leads receipt-freeness property.

$\mathcal{F}_{\text{TALLY}}$

With sender $P_1, \cdots, P_n$, tally server $T_1, \cdots, T_k$, administrator A and simulator $\mathcal{S}$. $L_C = \{C_1, \cdots, C_{N_C}\}$) is published.

Setup

1. Initialize $L_M = \phi$.

Key Generation

1. In the first activation, except to receive a message (`KeyGen`) from some sender P_i. Then,
   (a) Hand (`KeyGen`) to the simulator $\mathcal{S}$.
   (b) Receive a public key e and send e to P_i.

Encryption

1. When receiving $(\texttt{Encrypt}, m, e)$ from P_i,
   - if majority of tally servers is not corrupted then compute $c = e(m)$.
   - otherwise, $c = e(\mu)$, where μ is some fixed message.
2. Send c to P_i and record (m, c).

Decryption

1. $\gamma_i = 0 (i = 1, \cdots, N), \gamma_{inv} = 0$.
2. Upon receiving $(\texttt{Tally}, L)$ from T_i where $L = \{l_1, l_2, \cdots, l_{N'}\}$,
   (a) if it is first message from T_i then set $L_M \leftarrow L_M \cup \{i\}$.
   (b) send $(\texttt{Tally}, T_i)$ to $\mathcal{S}$
   (c) if $|L_M| \geq \frac{k}{2}$ then for $i = 1$ to N' do;
      - if (m, l_i) is stored for some m and $m = C_h$ for some $C_h \in L_C$ then $\gamma_h = \gamma_h + 1$.
      - otherwise, $\gamma_{inv} = \gamma_{inv} + 1$.
      - form $L' = \{\{\gamma_i\}_{i=1}^{N_C}, \gamma_{inv}\}$.
3. Upon receiving a message $(\texttt{Read}-\texttt{result})$ from P_i then if L' is computed as above return $(\texttt{Tally} - \texttt{result}, L')$ to P_i. Otherwise, ignore the message.

Figure 4.4: Functionality $\mathcal{F}_{\text{TALLY}}$

Chapter 5 Realization of functionalities

In this chapter, we introduce protocols that UC-realize the functionalities defined in Chapter 4. First, we in introduce a protocol that UC-realizes $\mathcal{F}_{\text{VOTE}}$ in ($\mathcal{F}_{\text{AMN}}$,$\mathcal{F}_{\text{TALLY}}$)-hybrid model. We then show how to realize $\mathcal{F}_{\text{AMN}}$ and $\mathcal{F}_{\text{TALLY}}$.

5.1 Realization of $\mathcal{F}_{\text{VOTE}}$

We introduce how to realize $\mathcal{F}_{\text{VOTE}}$ with a mix-net. First, we show a protocol π_{VOTE} that UC-realize $\mathcal{F}_{\text{VOTE}}$. For construction of π_{VOTE} we separate it into two parts, the mix part and the tally part. In the mix part, by using a mix-net ballots are shuffled and become anonymous. In the tally part, shuffled ballots are counted.

It may look like sufficient that voters by themselves count ballots because of anonymity of ballots. In such a case, however, a voter can prove that he votes an invalid ballot by writing a unique string because mix-net outputs shuffled ballots but the contents of the ballots can be read by anyone, which destroys the receipt-freeness property. It looks like unnecessary to conceal the contents of invalid ballots. Consider the following case, where Alice and Bob come forward as a candidate in an election and it is said that Alice has some advantage. Bob wants voters to vote invalid ballots rather than vote Alice (The best scenario for Bob is that purchased voters vote Bob. If anonymity of valid votes is achieved, it is impossible for Bob to convince the purchased voters really vote Bob. In this case, the second best scenario for Bob is that purchased voters vote an invalid vote). We, therefore, think that it is necessary to provide the tally part.

Voting systems must authenticate voters to prevent multiple voting and remove ballots from invalid parties that don't have the right to vote. We have two cases of authentication, authentication in the mix part or in the tally part. As shown in 5.3, in a mix-net each server have to prove that it honestly shuffles ballots according to the protocol by zero-knowledge proof. The computation of zero-knowledge proof is not efficient and voting systems should need many servers. It is desirable that the size of data that the mix-net operates is small as possible.

We, therefore, authenticate voters before ballots are shuffled. We introduce a mix-net protocol with authentication. All ballots counted in the tally part are from eligible voters. We will show functionalities assigned to the mix part and the tally part, and introduce protocols that UC-realize the two functionalities in chapter 5.45.5.3.

5.2 π_{VOTE}

In this chapter, we introduce a protocol π_{VOTE} that UC-realizes $\mathcal{F}_{\mathrm{VOTE}}$ in $(\mathcal{F}_{\mathrm{AMN}}, \mathcal{F}_{\mathrm{TALLY}})$-hybrid model. Note that, in hybrid model, parties has the interface with some functionalities during the protocol.

An outline of π_{VOTE} is as follows. See Figure 5.1 for detail. In the protocol, first, all voters send their ballots to $\mathcal{F}_{\mathrm{TALLY}}$. $\mathcal{F}_{\mathrm{TALLY}}$ returns the encrypted ballots. When voters receive the encrypted ballots, they send the ballots to $\mathcal{F}_{\mathrm{AMN}}$. $\mathcal{F}_{\mathrm{AMN}}$ has a eligible voter list, authenticates each voter, and shuffles the ballots only from eligible voters. $\mathcal{F}_{\mathrm{AMN}}$ outputs the list of the ballots that are anonymous. If a party sends a shuffled encrypted ballots to $\mathcal{F}_{\mathrm{TALLY}}$, $\mathcal{F}_{\mathrm{TALLY}}$ decrypts the ballots, counts them and outputs the result. The output is the list of the number of ballots that each candidate take, and the number of invalid ballots. The contents of invalid ballots are not published, which prevent the trade of invalid ballots.

π_{VOTE}

With voter $P_1, \cdots, P_n$, vote server $V_1, \cdots, V_k$.

voter P_i

1. When it takes as input $(\texttt{Vote}, m_i)$, send $(\texttt{Encrypt}, m_i)$ to $\mathcal{F}_{\mathrm{TALLY}}$.
2. Upon received $(\texttt{Encrypted}, M_i)$ from $\mathcal{F}_{\mathrm{TALLY}}$, send $(\texttt{Send}, M_i)$ to $\mathcal{F}_{\mathrm{AMN}}$.
3. When it takes as input $(\texttt{Read}-\texttt{result})$, send$(\texttt{Read})$ to $\mathcal{F}_{\mathrm{TALLY}}$.
4. Upon received $(\texttt{Tally}-\texttt{result}, L_{tally})$, output L_{tally}.

vote server V_j

1. When it takes as input $(\texttt{Start})$, send $(\texttt{Run})$ to $\mathcal{F}_{\mathrm{AMN}}$.
2. Upon received $(\texttt{List}, L')$ from $\mathcal{F}_{\mathrm{AMN}}$, send $(\texttt{Tally}, L')$ to $\mathcal{F}_{\mathrm{TALLY}}$.

Figure 5.1: Protocol π_{VOTE}

About π_{VOTE}, the following theorem holds;

Theorem 1. Assuming that more than half of vote servers are honest, π_{VOTE} UC realizes $\mathcal{F}_{\mathrm{VOTE}}$ for static adversaries in ($\mathcal{F}_{\mathrm{AMN}}$,$\mathcal{F}_{\mathrm{TALLY}}$)-hybrid model.

Proof: Let $\mathcal{H}$ be an static adversary in the hybrid model. We construct simulator $\mathcal{S}$ that runs $\mathcal{H}, \mathcal{F}_{\mathrm{AMN}}$ and $\mathcal{F}_{\mathrm{TALLY}}$ as black-boxes.

1. When $\mathcal{S}$ receives $(\texttt{Vote}, P_i)$, if P_i is not corrupted and $P_i \in L_p$ then $\mathcal{S}$ sends $(P_i, \texttt{Send})$ to $\mathcal{H}$ via $\mathcal{F}_{\mathrm{AMN}}$.
2. When $\mathcal{S}$ receives $(\texttt{Start}, V_j)$ from $\mathcal{F}_{\mathrm{VOTE}}$, if V_j is not corrupted then $\mathcal{S}$ sends $(\texttt{Tally}, V_j)$ to $\mathcal{H}$ via $\mathcal{F}_{\mathrm{TALLY}}$.
3. When $\mathcal{H}$ corrupted some party, $\mathcal{S}$ corrupts a corresponding dummy party and allows messages from environment $\mathcal{Z}$ hereafter.

It is clear that $\mathcal{S}$ simulates for $\mathcal{H}$ under the assumption that more than half of vote severs are honest. □

5.3 Realization of $\mathcal{F}_{\mathrm{AMN}}$

Wikström defined $\mathcal{F}_{\mathrm{MN}}$ and introduced a protocol that UC-realizes $\mathcal{F}_{\mathrm{MN}}$ (hereafter we call the protocol π_{MN}) [22]. We show π_{MN} in Figure 5.25.3.

π_{MN} (sender part)
With sender $P_1, \cdots, P_n$ and mix-server $M_1, \cdots, M_k$.
Sender P_i
1. Wait for $(\texttt{Publickeys}, y_1, \cdots, y_k)$ from $\mathcal{F}_{\mathrm{KG}}$ and compute $y = \Pi_{\iota_1}^{k} y_\iota$.
2. Wait for an input $(\texttt{Send}, m_i), m_i \in G_q$. Then choose $r_i \in G_q$ randomly and compute $(u_i, v_i) = E_y(m_i, r_i)$.
3. Hand $(\texttt{Prover}, (g, y, u_i, v_i), r_i)$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
4. Hand $(\texttt{Write}, (u_i, v_i))$ to $\mathcal{F}_{\mathrm{BB}}$.

Figure 5.2: Protocol π_{MN} (sender part)

In [22], Wikström proved the following proposition [22]. DDH-assumption is the assumption that a certain computational problem within a circle group is hard. We show formal description of the assumption in Figure 5.4.

Proposition 1. *π_{MN} UC-realizes $\mathcal{F}_{\mathrm{MN}}$ in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{KG}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}})$-hybrid model with respect to static adversaries that corrupt less than $\frac{k}{2}$ mix-servers under the DDH-assumption.*

$\mathcal{F}_{\mathrm{BB}}$ is a functionality of bulletin board. Anyone can read messages written on the board, but cannot rewrite or delete them. $\mathcal{F}_{\mathrm{KG}}$ is a functionality that generates keys that are needed to shuffle and recover messages. $\mathcal{F}_{\mathrm{ZK}}^{R}$ is a functionality of zero-knowledge proof about a relation R. R_{C} is a relation between plaintexts and ciphertexts, and R_{DS} is a relation between ciphertexts before shuffling and them after shuffling. We present $\mathcal{F}_{\mathrm{BB}}$, $\mathcal{F}_{\mathrm{KG}}$, $\mathcal{F}_{\mathrm{ZK}}^{R}$ in Figure 5.5, 5.6, 5.7, respectively.

In π_{MN}, senders send messages to the first mix-server and each server re-encrypts and shuffles the list of messages, and then outputs the list to next server. The last mix-server decrypts all messages in the list and outputs the list of plain messages. Shuffling by each servers make the messages be anonymous. In [22], Wikström showed protocols that UC-realizes these functionalities.

As shown Figure 5.3, each mix-server proves that it works according to π_{MN} and verifies proofs of other mix-servers. The input messages encrypted by public keys of mix-servers are decrypted step by step and are finally completely decrypted. If a proof by some server M^* does not pass the verification, other parties request $\mathcal{F}_{\mathrm{KG}}$ to show the secret key of M^* and decrypts instead of M^*.

5.4 Proposed protocol

We extend π_{MN} and introduce a protocol π_{AMN} that UC-realizes $\mathcal{F}_{\mathrm{AMN}}$. Indeed, we add an authentication mechanism to π_{MN}. We have to authenticate voters and remove ballots from invalid voters before the ballots are mixed for decrease of complexity.

As shown in Figure 5.3, in π_{MN} mix-servers form some set of senders (at step 5). We modify this part as follows. Mix-server M_j form the list L_* with restriction that L_* consists of the messages from the eligible voters. We assume that a list of eligible voters is published and all mix-servers knows the list, which does not affect the security of voting systems. In Figure 5.8, we present

the modified part of π_{AMN}.

It is clear that π_{AMN} UC-realizes $\mathcal{F}_{\mathrm{AMN}}$ in $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{KG}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}})$-hybrid model. We assume that there exist a list of eligible voter, L_P on $\mathcal{F}_{\mathrm{BB}}$ and more than half of mix server are honest. The difference between $\mathcal{F}_{\mathrm{AMN}}$ and $\mathcal{F}_{\mathrm{MN}}$ is only authentication of voters. Authentication is carried by a list and the list is published. We, therefore, can construct a simulator $\mathcal{S}$ such that any $\mathcal{Z}$ cannot distinguish the ideal world with $\mathcal{F}_{\mathrm{AMN}}$ and the real world with π_{AMN} by using $\mathcal{S}^*$, where $\mathcal{S}^*$ is a simulator such that any $\mathcal{Z}$ cannot distinguish the ideal world with $\mathcal{F}_{\mathrm{MN}}$ and the real world with π_{MN}.

5.5 Realization of $\mathcal{F}_{\mathrm{TALLY}}$

In this chapter, we show how to realize $\mathcal{F}_{\mathrm{TALLY}}$. $\mathcal{F}_{\mathrm{TALLY}}$ has two roles; encryption of ballots from voters, and publishing the result of counting the encrypted voters. Notice that $\mathcal{F}_{\mathrm{TALLY}}$ does not publish the contents of invalid ballots. To realize $\mathcal{F}_{\mathrm{TALLY}}$, we use the idea of ElGamal encryption and threshold decryption. First, we will introduce these techniques and then show a protocol π_{TALLY} that UC-realizes $\mathcal{F}_{\mathrm{TALLY}}$.

5.5.1 ElGamal Encryption

The ElGamal encryption system is introduced by ElGamal whose security is based on the difficulty of discrete logarithm [10]. We present the idea of ElGamal encryption in Figure 5.9.

ElGamal encryption has the following properties. Let $Enc(m, r)$ a ciphertexts of m randomized by r.

- $Enc(m/m', r/r') = (g^{r-r'}, m/m'y^{r-r'})$
- $Enc(m^a, ar) = (g^{ar}, m^a y^{ar})$

By these properties, we can use the following technique; Let $(u_1, e_1) = Enc(m_1, r_1)$ be a ciphertext of m_1 randomized by r_1, (u_2, e_2) be a ciphertext of m_2 randomized by r_2. Compute $(u, e) = (u_1/u_2, e_1/e_2) = (g^{r_1-r_2}, m_1/m_2 y^{r_1-r_2})$. We obtain $m = m_1/m_2$ by decrypting (u, e). Here, if $m_1 = m_2$ then $m = 1$, otherwise m is some value. Then, compute (u^R, e^R), where R is a random value. By decryption, we obtain $m^R = (m_1/m_2)^R$. If $m_1 = m_2$ then $m = 1$, otherwise m^R is random value because R is random. Consider the case that m_1 is a ballot

and m_2 is one of candidate of election. If m_1 is a valid ballot, that is, the name of some candidate, then $m^R = 1$ for some m_2. Otherwise, in the case that m_1 is invalid, m^R is random, thus no one can obtain the content of m_1. It prevents someone from buying invalid ballots.

ElGamal Encryption is IND-CPA (i.e. indistinguishable against chosen plaintext attack) but not IND-CCA (i.e. indistinguishable against chosen ciphertext attack). From a ciphertext of $\hat{m}$, $(\hat{u}, \hat{e})$, anyone can easily obtain a ciphertext of $2\hat{m}$, $(\hat{u}, 2\hat{e})$. In the voting system, $\hat{m}$ denotes a candidate so that $2\hat{m}$ has no meaning. A ciphertext of $2\hat{m}$ is counted as an invalid ballots and the contents of invalid ballots are not published. That is, CCA attacks don't destroy the security of voting systems. ElGamal Encryption as IND-CPA is enough to encrypt ballots.

5.5.2 Threshold decryption

In standard encryption systems, a party who has a secret key can decrypt by itself ciphertexts encrypted by a corresponding public key. On the other hand, in threshold decryption systems, a secret key is divided into some secret shares and each party has one share. For a threshold t, only when t parties jointly work ciphertexts are decrypted. $t-1$ parties cannot get any information about plaintexts. In standard encryption system, if a secret key is stolen by an adversary, the adversary can decrypts any messages encrypted by public key according to the stolen secret key. In threshold decryption systems, even if a secret share is stolen by an adversary, he cannot get any information by itself and the secrecy of encrypted documents are maintained.

Tamura, Shiotsuki and Miyaji introduced a threshold decryption system of ElGamal [21]. We use this technique and prevent malicious parties from working jointly to obtain the context of ballots. Unless more than t tally servers are corrupted, receipt-freeness is guaranteed.

5.5.3 Proposed protocol

In this chapter, we introduce a protocol π_{TALLY} that UC-reales $\mathcal{F}_{\text{TALLY}}$. We use ElGamal Encryption with threshold decryption. We consider threshold $t = \frac{k}{2}$ in the use of π_{TALLY}. We assume that there exists Administrator A that always works honestly and is never corrupted. We present π_{TALLY} in Figure 5.10

Assume that a list of eligible voter¡s L_P and a list of candidates L_C is published and any party can read them. As setup, A generate a secret key, a corresponding public key and secret shares of the secret key. A then publishes the public key and sends secret shares to tally servers. Moreover, A encrypts with the secret key all candidates of L_C. Let the resulting list L'_C. A voter encrypts a ballot by ElGamal with the public key, and then votes the encrypted ballots.

When a voter want to count a list of encrypted and mixed ballots L, he sends L to all tally servers. More than threshold tally servers jointly decrypt L and count ballots. Finally, when the voter obtains the result of counting, he outputs it.

Theorem 2. *π_{TALLY} UC-realizes $\mathcal{F}_{\text{TALLY}}$ with static adversaries under the assumption that more than half of tally servers are honest.*

Proof: We construct a simulator $\mathcal{S}$ that no environment $\mathcal{Z}$ can tell with non-negligible probability whether it interacts with the ideal world or with the real world. $\mathcal{S}$ locally runs senders, tally-servers and administrator as a simulation, and executes π_{TALLY}. $\mathcal{S}$ proceeds as follows.

1. When $\mathcal{S}$ receives a message (`KeyGen`) from $\mathcal{F}_{\text{TALLY}}$, it activates administrator, obtain a secret key and a public key and send the public key to $\mathcal{F}_{\text{TALLY}}$.
2. When $\mathcal{S}$ receives $(\texttt{Run}, M_i)$ from $\mathcal{F}_{\text{TALLY}}$, it sends (Run) to M_i in the simulation.
3. When S receives a message $(\texttt{Encrypt}, e', |m|)$ from $\mathcal{F}_{\text{TALLY}}$, it first verifies $e' = e$.
   - If so, it computes $c = Enc_e(0^{|m|}, r)$ for random r and returns c to $\mathcal{F}_{\text{TALLY}}$.
   - Otherwise, it returns error message to $\mathcal{F}_{\text{TALLY}}$.
4. When S receives a message $(\texttt{Decrypt}, c)$ from $\mathcal{F}_{\text{TALLY}}$, if it has already sent (`Run`) to more than $\frac{k}{2}$ tally-servers, then it computes $m = Dec_x(c)$ and returns m to $\mathcal{F}_{\text{TALLY}}$.

We show that S described above completes the simulation. It is clear that if π_{AMN} runs honestly then no environment can distinguish. If $\mathcal{S}$ receives an

instruction from $\mathcal{Z}$ to corrupt some party, it corrupts the party in the simulation and follows messages from $\mathcal{Z}$ hereafter. When $\mathcal{Z}$ require corrupted parties to hand some data about encrypted message during the case that less than $\frac{k}{2}$ tally servers are corrupted, $\mathcal{S}$ hands random value to $\mathcal{Z}$ as corrupted parties. Because of the property of threshold decryption, $\mathcal{Z}$ cannot get any information about the encrypted message so that $\mathcal{Z}$ cannot distinguish.

The deference between the ideal world and the simulation is only that we must encrypt fake messages in the simulation as shown in Figure 5.10. Because of the secrecy of ElGamal encryption, however, there exists no environment that can distinguish from the encrypted messages. □

π_{MN} (mix-server part)

Mix-server M_j

1. Choose $x_j \in Z_q$ randomly and hand $(\texttt{MyKey}, x_j, g^{x_j})$ to $\mathcal{F}_{\mathrm{KG}}$
2. Wait for $(\texttt{Keys}, x_j, y_1, \cdots, y_k)$ from $\mathcal{F}_{\mathrm{KG}}$, compute $h_l = \Pi_{j=l}^{k} y_j$ for $l = 1, \cdots, k$ and set $y = h_1$.
3. Wait for an input $(\texttt{Run}, L_C)$, and then hand $(\texttt{Write}, \texttt{Run})$ to $\mathcal{F}_{\mathrm{BB}}$.
4. Wait until at least half different mix-servers has written `Run` on $\mathcal{F}_{\mathrm{BB}}$, and let the last entry of this type be $(c_{run}, M_i, \texttt{Run})$.
5. Form the list $L* = \{(u_\gamma, v_\gamma)\}_{\gamma \in I_*}$, for some index set I_*, from the set of entries on $\mathcal{F}_{\mathrm{BB}}$ on the form $(c, P_\gamma, (u_\gamma.v_\gamma))$, where $0 \geq c < c_{run}, \gamma \in \{1, \cdots, N\}$, and $u_\gamma, v_\gamma \in G_q$.
6. For each $\gamma \in I_*$ do the following,
   (a) Hand $(\texttt{Question}, P_\gamma, (g, y, u_\gamma, v_\gamma))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
   (b) Wait for $(\texttt{Verifier}, P_\gamma, b_\gamma)$ from $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.

   Then form $L_0 = \{(u_{0,i}, v_{0,i})\}_{i=1}^{N'}$ consisting of pairs (u_γ, v_γ) such that $b_\gamma = 1$.
7. For $l = 1, \cdots, k$ do;
   (a) If $l \neq j$, then do;
      i. Wait until an entry $(c, M_l, (\texttt{List}, L_l))$ appears on $\mathcal{F}_{\mathrm{BB}}$.
      ii. Hand $(\texttt{Question}, M_l, (g, h_{l+1}, y_l, L_{l-1}, L_l))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}}$, and wait for $(\texttt{Verifier}, M_l, b_l)$ from $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}}$.
      iii. If $b_\gamma = 0$, then hand $(\texttt{Recover}, M_l)$ to $\mathcal{F}_{\mathrm{KG}}$, and wait for $(\texttt{Recovered}, M_l, x_l)$ from $\mathcal{F}_{\mathrm{KG}}$. Then define$L_l = \{D_{x_j}(u_{l-1,i}, v_{l-1,i})\}_{i=1}^{N'}$.
   (b) If $l = j$, then choose $r_{j,i} \in Z_q$ and $\pi_j \in \Sigma_{N'}$ randomly, and compute $L_j = \{(g^{r_{j,i}} u_{j-1,\pi_j^{-1}(i)}, h_{j+1}^{r_{j,i}} \frac{v_{j-1,\pi_j^{-1}(i)}}{u_{j-1,\pi_j^{-1}(i)}^{x_j}})\}_{i=1}^{N'}$. Finally hand $(\texttt{Prover}, (g, h_{j+1}, y_j, L_{j-1}, L_j), (x_j, \pi_j, \{r_{j,i}\}_{i=1}^{N'}))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}}$, and hand $(\texttt{Write}, (\texttt{List}, L_j))$ to $\mathcal{F}_{\mathrm{BB}}$.
8. Sort $\{v_{k,i}\}_{i=1}^{N'}$ lexicographically to form a list L' and output $(\texttt{Output}, L')$.

Figure 5.3: Protocol π_{MN} (mix-server part)

DDH-assumption

There exists no probabilistic polynomial algorithm $\mathcal{A}$ which can distinguish $(p, g, g^x \bmod p, g^y \bmod p, g^{xy} \bmod p)$ and $(p, g, g^x \bmod p, g^y \bmod p, g^r \bmod p)$ for random r.

Figure 5.4: DDH-assumption

$\mathcal{F}_{\mathrm{BB}}$

With party $P_1, \cdots, P_N$ and simulator $\mathcal{S}$.

1. $c = 0$
2. Upon receiving $(\texttt{Write}, m_i)$ from P_i, let $c = c + 1$, store $(c, P_i.m_i)$ and send $(\texttt{Write}, c, P_i, m_i)$ to $\mathcal{S}$.
3. Upon receiving $(\texttt{Read}, c)$ from P_j, check that (c, P_i, m_i) is stored;
   - If so, send $(\texttt{Read}, P_j, c)$ to $\mathcal{S}$ and send $(\texttt{Read}, c, P_i, m_i)$ to P_j.
   - otherwise, send $(\texttt{NoRead}, P_j, c)$ to $\mathcal{S}$ and send $(\texttt{Noread}, c)$ to P_j.

Figure 5.5: Functionality $\mathcal{F}_{\mathrm{BB}}$

$\mathcal{F}_{\mathrm{KG}}$

With sender $P_1, \cdots, P_n$, mix-server $M_1, \cdots, M_k$ and simulator $\mathcal{S}$.

1. $J_j = \phi$ for $j = 1, \cdots, k$.
2. Until $|J_0| = k$ wait for $(\texttt{Mykey}, x_j, y_j)$ from M_j. Set $J_0 \leftarrow J_0 \cup \{j\}$.
3. Hand $(\texttt{Publickeys}, y_1, \cdots, y_k)$ to $\mathcal{S}$ and all senders, and hand $(\texttt{Keys}, x_j, y_1, \cdots, y_k)$ to $M_j (j = 1, \cdots, k)$.
4. If $(\texttt{Recover}, M_l)$ is received from M_j, set $J_l \leftarrow J_l \cup \{j\}$. If $|J_l| \geq \frac{k}{2}$, then hand $(\texttt{Recovered}, M_l, x_l)$ to $\mathcal{S}$ and all mix-servers, and otherwise hand $(M_j, \texttt{Recover}, M_l)$ to $\mathcal{S}$.

Figure 5.6: Protocol $\mathcal{F}_{\mathrm{KG}}$

$\mathcal{F}_{\text{ZK}}^{R}$

With prover $P_1, \cdots, P_N$, verifier $M_1, \cdots, M_k$ and simulator $\mathcal{S}$.

1. Upon receipt of $(\texttt{Prover}, x, w)$ from P_i, store w under the tag (P_i, x), and hand $(P_i, \texttt{Prover}, x, R(x, w))$ to $\mathcal{S}$. Ignore further messages from P_i.
2. Upon receipt of $(\texttt{Question}, P_i, x)$ from M_j, let w be the string stored under the tag (P_i, x), and hand $(M_j, \texttt{Verifier}, P_i, x, R(x, w))$ to $\mathcal{S}$ and hand $(\texttt{Verifier}, P_i, R(x, w))$ to M_j.

Figure 5.7: Functionality $\mathcal{F}_{\text{ZK}}^{R}$

π_{AMN} (partial)

A list $L_P = \{l_1, \cdots, l_{N_P}\}$ is published.

Mix-server M_j

1. Choose $x_j \in Z_q$ randomly and hand $(\texttt{MyKey}, x_j, g^{x_j})$ to $\mathcal{F}_{\text{KG}}$
2. Wait for $(\texttt{Keys}, x_j, y_1, \cdots, y_k)$ from $\mathcal{F}_{\text{KG}}$, compute $h_l = \Pi_{j=l}^{k} y_j$ for $l = 1, \cdots, k$ and set $y = h_1$.
3. Wait for an input $(\texttt{Run}, L_C)$, and then hand $(\texttt{Write}, \texttt{Run})$ to $\mathcal{F}_{\text{BB}}$.
4. Wait until at least half different mix-servers has written `Run` on $\mathcal{F}_{\text{BB}}$, and let the last entry of this type be $(c_{run}, M_i, \texttt{Run})$.
5. Form the list $L* = \{(u_\gamma, v_\gamma)\}_{\gamma \in I_*}$, for some index set I_*, from the set of entries on $\mathcal{F}_{\text{BB}}$ on the form $(c, P_\gamma, (u_\gamma.v_\gamma))$, where $0 \leq c < c_{run}, \gamma \in \{1, \cdots, N\}$, $u_\gamma, v_\gamma \in G_q$, and $P_\gamma \in L_P$.
6. The rest is the same with π_{MN}

Figure 5.8: Protocol π_{AMN}

ElGamal Encryption

- p, q : prime numbers such that $q|p-1$.
- $g \in Z_p^*$: a generator which order is q.
- x : a secret key
- public keys are $g, y = g^x \bmod p$

Encryption for message m

1. Choose $r \in Z_p^*$ randomly.
2. Output $(u, e) = (g^r, my^r)$ as a ciphertext.

Decryption

1. Compute $m = \frac{u}{e^x} \bmod p$.

Figure 5.9: ElGamal Encryption

π_{TALLY}

With sender $P_1, \cdots, P_n$, tally-server $T_1, \cdots, T_k$ and administrator A. $L_C = \{C_1, \cdots, C_{N_C}\}$ is a list of candidates.

Setup

1. A randomly generates $x \in Z_q^*, g \in Z, r'$, publish $g, y = g^x$ as ElGamal public keys.
2. A encrypts each elements C_j of L_C by ElGamal with r', that is compute $(u'_j, e'_j) = (g^{r'}, c_j y^{r'})$. Let the encrypted list $L' = \{(u'_1, e'_1), \cdots, (u'_{N_C}, e'_{N_C})\}$.
3. A generates k shares of x, s_j $(j = 1, \cdots, k)$ and send s_j to T_j $(j = 1, \cdots, k)$.

Encryption (Case that P_i encrypts v_i)

1. P_i randomly choose $r_i \in Z$ and computes $(u_i, e_i) = (g^{r_i}, v_i y^{r_i})$.
2. P_i send (u_i, e_i) to all tally-servers.
3. Let a subset of $k - t + 1$ tally-servers that are activated by `Run` $I_\tau = \{\tau_1, \cdots, \tau_{k-t+1}\}$. Tally-servers in I_τ computes (u_i^R, e_i^R) and $(\hat{u}_\iota^R, \hat{e}_\iota^R)$ $(\iota = 1, cdots, N_C)$.
4. P_i receives (u_i^R, e_i^R) and output it.

Tally

1. P_i send to all tally-servers a list $L = \{l_1, \cdots, l_n\}$, where each elements are votes encrypted as above.
2. t tally-serves jointly execute as follows; for $\alpha = 1$ to n;
   (a) for $\beta = 1$ to N_C;
   - set $f = 0$.
   - compute $(u_{\alpha,\beta}, e_{\alpha,\beta}) = (u_\alpha / hatu_\beta, e_\alpha / \hat{e}_\beta)$.
   - decrypt $(u_{\alpha,\beta}, e_{\alpha,\beta})$ and get $w_{\alpha,\beta} = (v_\alpha / \hat{v}_\beta)^R$.
   - if $w_{\alpha,\beta} = 1$ then set $\gamma_\beta = \gamma_\beta + 1$ and $f = 1$.

   (b) if $f = 0$ then $\gamma_{inv} = \gamma_{inv} + 1$.
3. Send $L_O = \{\gamma_1, \cdots, \gamma_{N_C}, \gamma_{inv}\}$ to P_i.
4. P_i outputs L_O.

Figure 5.10: Protocol π_{TALLY}

Chapter 6 The security of blind signatures

Construction of voting by a mix-net technique is intuitive and easy to understand. As shown above, mix-net techniques, however, require much complexity to prove that mix-servers work honestly by zero knowledge proof. Parties checks validity of voting by verifying the proofs. For large systems that a lot of parties join in, the complexity is very serious.

In blind signatures, validity is checked by signature verification, which does not require the much complexity like as mix-net techniques. To construct voting systems by blind signature seems more realistic. Thus, it is significant to evaluate blind signatures in the UC framework.

Canneti showed the ideal functionality of digital signatures [4] and evaluate the security of digital signatures in the UC framework. No functionality of blind signatures, however, have ever been defined and the security of blind signatures in the UC framework is not evaluated. In this research, we define an ideal functionality of blind signatures. Before formulate the definition, we show the traditional security of blind signatures formulated by Juels, Lindell and Ostrovsky. Additionally, we formulate the stronger and more appropriate security than that of Juels et. al.'s.

6.1 The definition of blind signatures

Juels, Luby and Ostrovsky formally defined the blind signature scheme [14]. In their definition, a blind signature scheme is the four-tuple (Signer, User, *Gen, Verify*). $Gen(1^k)$ is a probabilistic polynomial time algorithm which outputs a pair of public key and secret key, (pk, sk). Signer and User are polynomially-bounded probabilistic interactive Turing machines, where both machines have the following tapes:

- read-only input tape
- write-only output tape
- read/write work tape
- read-only random tape
- read-only communication tape

- write-only communication tape

A public key pk produced by *Gen* is given to Signer and User as a common input. Additionally Signer is given a corresponding key sk, and User is given a message m. The length of all inputs must be polynomial in the security parameter 1^k. They interact with each other according to the protocol. After the computation, Signer outputs either *completed* or *non-completed*, and User outputs either *fail* or $\sigma(m)$. When verifying the signature, *Verify*$(pk, m, \sigma(m))$ is a signature verification algorithm which outputs either *accept* or *reject*. It is required that for any message m and random choice of *Gen* if both Signer and User follow the protocol then Signer always outputs *completed* and *Verify* always outputs *accept*.

6.2 Security properties

Jules, Luby and Ostrovsky formulated the security of blind signatures: blindness and non-forgeability. They defined that;

Definition 1. A blind signature scheme is *secure* if for all constants c and for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a security parameter $k_{c,\mathcal{A}}$ such that for all $k > k_{c,\mathcal{A}}$ the following two properties hold:

Blindness Let $b \in_R \{0, 1\}$, where b is kept from $\mathcal{A}$. $\mathcal{A}$ executes the following experiment:

1. $(pk, sk) \leftarrow Gen(1^k)$.
2. $\{m_0, m_1\} \leftarrow \mathcal{A}(1^k, pk, sk)$.
3. We denote by $\{m_b, m_{1-b}\}$ the same two documents $\{m_0, m_1\}$, ordered according to the value of b, which is still kept from $\mathcal{A}$. $\mathcal{A}(1^k, pk, sk, m_0, m_1)$ engages in two parallel interactive protocols, the first with User(pk, m_b) and the second with User(pk, m_{1-b}).
4. If the first User outputs on her private tape $\sigma(m_b)$ and the second User outputs $\sigma(m_{1-b})$ then $\mathcal{A}$ is given as an additional input $\{\sigma(m_b), \sigma(m_{1-b})\}$ ordered according to the corresponding (m_0, m_1) order.
5. $\mathcal{A}$ outputs bit $\tilde{b}$.

The probability, taken over the choice of b, over coin-flips of *Gen*, the coin-flips of $\mathcal{A}$, and coin-flips of both users in step 3, that $\tilde{b} = b$ is at most

$\frac{1}{2} + \frac{1}{k^c}$.

Non-forgeability $\mathcal{A}$ executes the following experiment:

1. $(pk, sk) \leftarrow Gen(1^k)$.
2. $\mathcal{A}(pk)$ engages in polynomially many adaptive, parallel and arbitrarily interleaved interactive protocols with polynomially many copies of Signer, where $\mathcal{A}$ decides in an adaptive fashion when to stop. Let l denote the number of executions, where Signer outputs *completed* at the end of step 2.
3. $\mathcal{A}$ outputs a collection $\{(m_1, \sigma(m_1)), \cdots, (m_j, \sigma(m_j))\}$ subject to the constraint that $\forall i_1, i_2\ m_1 \neq m_2 (i_1, i_2 \in \{1, \cdots, j\}, i_1 \neq i_2)$, and that that all $(m_i, \sigma(m_i))$ for $1 \leq i \leq j$ are *accepted* by *Verify*$(pk, m_i, \sigma(m_i))$.

It follows that the probability, taken over, coin-flips of *Gen*, the coin-flips of $\mathcal{A}$ and the coin-flips of Signer, that $j > l$ is at most $\frac{1}{k^c}$.

Notice that we here introduced non-forgeability as normal non-forgeability (i.e., not strong non-forgeability) while in [14], strong non-forgeability is defined as one of the security properties. Jules, Luby and Ostrovsky proved the following proposition [14].

Proposition 1. *Assume that one-way trapdoor permutations exist. Then there exists a polynomial-time blind signature scheme, secure against an adaptive interleaved chosen-message attack.*

6.3 Stronger security definition

In the definition of security properties in Chapter 6.2, adversaries use *pk* output by *Gen*. Adversaries cannot choose keys that may be useful for attacks. Usually, however, a signer generates keys in the plain model so that it is not guaranteed that *pk* is always generated honestly ('in the plain model' means that there exists no assumption except for authenticated channels. The plain model, thus, is close to our real life world).

We have to define stronger security against the case that a malicious signer generates keys. It is more appropriate for many applications. Indeed, we define Strong Blindness as follows. Non-forgeability is the same as the security in Chapter 6.2

Strong Blindness Let $b \in_R \{0,1\}$, where b is kept from $\mathcal{A}$. $\mathcal{A}$ executes the following experiment:

1. $\{pk, m_0, m_1\} \leftarrow \mathcal{A}(1^k)$
2. We denote by $\{m_b, m_{1-b}\}$ the same two documents $\{m_0, m_1\}$, ordered according to the value of b, which is still kept from $\mathcal{A}$. $\mathcal{A}(1^k, pk, m_0, m_1)$ engages in two parallel interactive protocols, the first with User(m_b, pk) and the second with User(m_{1-b}, pk).
3. If the first User outputs on her private tape $\sigma(m_b)$ and the second User outputs $\sigma(m_{1-b})$ then $\mathcal{A}$ is given as an additional input $\{\sigma(m_b), \sigma(m_{1-b})\}$ ordered according to the corresponding (m_0, m_1) order.
4. $\mathcal{A}$ outputs a bit $\tilde{b}$

The probability, taken over the choice of b, the coin-flips of $\mathcal{A}$, and coin-flips of both users in step 3, that $\tilde{b} = b$ is at most $\frac{1}{2} + \frac{1}{k^c}$.

Definition 2. A blind signature scheme is *SB-secure* if for all constants c and for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a security parameter $k_{c,\mathcal{A}}$ such that for all $k > k_{c,\mathcal{A}}$ Stronger Blindness and Non-forgeability hold.

Lemma 1. *If a blind signature is SB-secure then it is secure (in the sense of Juels et al.).*

Proof: We show that if there exists an adversary that breaks security we can construct another adversary that breaks SB-security. Because non-forgeability is the same for both types of security, we concentration on blindness, that is, we show that we can construct an adversary that breaks strong blindness by using another adversary that breaks (normal) blindness.

Assume that $\mathcal{A}^*$ breaks blindness. We can construct $\mathcal{A}$ that breaks strong blindness as follows. $\mathcal{A}$ locally runs $\mathcal{A}^*$. $\mathcal{A}$ gives inputs to $\mathcal{A}^*$ and reads outputs of $\mathcal{A}^*$. In the strong blindness experiment, before output $\{pk, m_0, m_1\}$ $\mathcal{A}$ randomly generates sk and pk, and sends them to $\mathcal{A}^*$ instead of Gen in the blindness experiment. When $\mathcal{A}^*$ outputs $\{m_0, m_1\}$, $\mathcal{A}$ outputs $\{pk, m_0, m_1\}$. $\mathcal{A}$ delivers messages from Users to $\mathcal{A}^*$ and vice versa during the interaction with them. When $\mathcal{A}^*$ outputs $\tilde{b}$ $\mathcal{A}$ also outputs it. If $\mathcal{A}$ succeeds the guess $\mathcal{A}^*$ always succeeds the guess. Therefore, the probability that $\mathcal{A}$ succeeds in this guess is

the same as that $\mathcal{A}^*$'s guess succeeds. Therefore if $\mathcal{A}^*$ breaks blindness then $\mathcal{A}$ breaks strong blindness. $\square$

Chapter 7 An ideal functionality of blind signatures

In this chapter we define an ideal functionality of blind signatures. The main role of the functionality is to act as an anonymous message storage. It stores pairs of messages and signatures. Each signature can be considered as a tag of a corresponding message. At each signature generation, a signer can know the signature but cannot know the message signed. Upon receiving a request for the verification of a signature and a message, the functionality checks if the pair of the message and the signature is already stored. Canetti defined the functionality of digital signature, $\mathcal{F}_{\mathrm{SIG}}$ [4]. We extend $\mathcal{F}_{\mathrm{SIG}}$ and define the functionality of blind signatures

The functionality of blind signatures is specified by a signature scheme. Key generation and signature choosing are depend on the signature scheme. The functionality provides a verification key and at the request to issue a signature it chooses a value from the distribution of the output of signature scheme. Thus, the behavior of the functionality depends on the signature scheme used in it.

We describe $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$ in Figure 7.1, which is the ideal functionality of blind signatures parameterized by a signature scheme Π. In Figure 7.1, Gen and Σ is specified by Π. Σ is a distribution of signatures in Π. Here, we introduce an outline of $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$. At the request of Signer, it sends a verification key to Signer and all Users. When User sends $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$ a request with a verification key to issue a signature, $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$ ask the simulator $\mathcal{S}$. When $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$ receives a message from $\mathcal{S}$ to generate a signature, it randomly chooses σ from Σ and sends it to User. At the request of a verifier, it returns an output according to the following scenario. In Figure 7.1, 2-(a) is the scenario where v' and (m', σ') are valid. Blind signatures must guarantee that if a pair of a message and a document is generated correctly, the pair always passes the verification. 2-(b) occurs when Signer is not corrupted and the document has not been signed before. It must be guaranteed that a fake pair, that is, a signature is forged, the pair does not pass the verification with overwhelming probability. 2-(c) means that when some result for this input is already stored, then $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$ must returns the same result

regardless of the signer's condition. 2-(d) occurs when Signer is corrupted and no result is stored. In this case $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$ returns the result decided by the simulator because Signer is corrupted and the simulator can control the verification result.

$\mathcal{F}_{\text{BSIG}}^{\Pi}$

With User $P_i (i = 1, \cdots, n)$, Signer Q, Simulator S.

- **Key Generation**
  1. In the first activation, expect to receive (`KeyGen`) from Q; upon receipt send it to S.
  2. Upon receiving (`Key`, v) from S, store (Q, v), send (`Verification Key`, v) to all User and Q.
- **Signature Generation**
  1. Upon receiving (`Sign`, m, v) from P_i, check that v is already stored. If so (`Request`, v, P_i) to S. Otherwise, send (`Reject`) to P_i.
  2. If (`Signature`, `Completed`) is received from S, randomly choose σ from $\Pi(m, v)$, store $(m, \sigma, v, 1)$, send (`Signature`, σ, m) to P_i, and send (`Completed`) to Q (Given pk and m, $\Pi(m, pk)$ is the random variable of outputs of honest User with input pk after interacting with honest Signer with input sk, where (pk, sk) is an output of $Gen(1^k)$ and the probability of $\Pi(m, pk)$ is over the randomness of User, Signer and Gen). Otherwise, send (`Fail`) to P_i and send (`Not-completed`) to Q.
- **Signature Verification**
  1. Upon receiving (`Verify`, m', σ', v') from P_j, send it to S.
  2. Assume (`Verified`, m', ϕ) is received from S.
     (a) if $v' = v$ and $(m', \sigma', v', 1)$ is stored, then set $f = 1$.
     (b) else if $v' = v$, m' is never signed and Q is not corrupted, then store $(m', \sigma', v', 0)$ and set $f = 0$.
     (c) else if (m', σ', v', f') is stored, then set $f = f'$.
     (d) else set $f = \phi$ and record (m', σ', v', ϕ)
  3. If $f = 1$ then send (`Accept`) to P_j, otherwise send (`Reject`) to P_j.

Figure 7.1: Ideal functionality of blind signatures

Chapter 8 Comparison of the security

In this chapter, we compare the UC security and the traditional security of blind signatures to analyze the security of blind signatures in the UC framework. We compare the UC security and the SB-security, and the security by Juels et. al.'s, respectively. We consider sets of protocols that are secure in each security class. We show that the UC security is stronger than the security by Juels et. al.'s in 8.1, and show that the UC security is stronger than the SB security in 8.2.

8.1 UC-secure blind signatures are not equivalent to secure blind signatures

In this chapter we show;

Theorem 3. *The class of UC-secure blind signatures is a proper subset of that of secure blind signatures for static adversaries, assuming one-way trapdoor permutations.*

To prove Theorem 3 we first show;

Lemma 2. *There exists a protocol that is secure but not UC-secure for static adversaries, assuming the one-way trapdoor permutations.*

Proof: We present an instance of the JLO protocol and show that it is not UC-secure. The JLO protocol uses *the two-party completeness theorem* for realizing blindness. This theorem was shown by Yao and Goldreich, Micali and Wigderson [23, 12]; they said that for any two parties A and B where A is given secret input x and B is given secret input y, and for any polynomial-time computable function $g(\cdot, \cdot)$ there exists a protocol for computing $g(x, y)$ such that nothing except for the output of the function is revealed to the parties. In [12], Goldreich, Micali and Wigderson showed at first that assuming the existence of a trapdoor permutation, any functionality can be securely realized by a protocol in the semi-honest adversary model. In the semi-honest adversary model, even if a party is corrupted it works according to a prescribed protocol and the adversary only has access to the state of corrupted parties. Next, they constructed *a protocol compiler* that transforms a protocol in the semi-honest model to work securely in the malicious adversary model. The compiler makes

each party prove, in zero-knowledge proof manner, that each message it sends was honestly made from its input, its random choice, and the messages it has received so far. It prevents a corrupted party from diverging from the protocol without being detected by other honest party and the adversary is limited to semi-honest behavior. Goldreich, Micali and Wigderson showed that zero-knowledge proof protocol can be constructed assuming one-way permutations [11].

We denote by Σ_1 the instance of the JLO protocol that uses as a black box the Blum's zero-knowledge proof protocol [2, 5]. The zero-knowledge protocol uses commitments that do not require common strings, namely that works in the plain model. We then construct Σ_2 as follows. We replace the commitment protocol in the zero-knowledge proof protocol in Σ_1 by $\mathcal{F}_{\mathrm{COM}}$, the ideal functionality of commitments defined in [5]. Here, we show that Σ_2 UC-realizes $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$. The JLO protocol uses the signature scheme of Naor and Yung. Thus, we let Π be the signature scheme of Naor Yung, and specify $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$ In [5], it was proved that the Blum's zero-knowledge proof protocol in which commitments protocols are replaced by $\mathcal{F}_{\mathrm{COM}}$ UC-realizes $\mathcal{F}_{\mathrm{ZK}}$, the ideal functionality of zero-knowledge proofs. Moreover, Canetti, Lindell, Ostrovsky and Sahai proved that by using $\mathcal{F}_{\mathrm{ZK}}$, any two-party functionality can be UC-realized [7]. They introduced *the universally composable protocol compiler* which is a UC version of the protocol compiler [12]. Now, it is guaranteed that if no party is corrupted and the execution runs honestly then any environment cannot distinguish. Thus, Σ_2 UC-realizes $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$.

Here, we assume that Σ_1 UC-realizes $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$. Then it holds that Σ_1 and Σ_2 are indistinguishable. Recall that the difference between Σ_1 and Σ_2 is the part of commitments; Σ_1 uses an actual commitment protocol and Σ_2 uses $\mathcal{F}_{\mathrm{COM}}$. Thus, the fact that Σ_1 and Σ_2 are indistinguishable means that the commitment protocol in the zero-knowledge proof protocol in Σ_1 UC-realizes $\mathcal{F}_{\mathrm{COM}}$. Canneti and Fischlin, however, proved that no two-party protocol can UC-realize $\mathcal{F}_{\mathrm{COM}}$ in the plain model [5]. From the contradiction we can say that Σ_1 does not UC-realize $\mathcal{F}_{\mathrm{BSIG}}^{\Pi}$. □

Next, we show;

Lemma 3. *If a blind signature protocol is UC-secure then it is also secure in the sense of Juels et. al..*

Proof: We show that if a protocol is not secure it is not UC-secure. Assume that a protocol is not secure, that is, there exists an adversary that breaks blindness or non-forgeability. First, consider the case that blindness is broken and denote by $\mathcal{A}^*$ the adversary that breaks blindness with the probability ε_1. We then construct an environment $\mathcal{Z}^*$ as follows. $\mathcal{Z}^*$ corrupts a signer and gets a verification key and a corresponding secret key (v^*, sk^*). $\mathcal{Z}^*$ then sends (v^*, sk^*) to $\mathcal{A}^*$. When $\mathcal{A}^*$ generates (m_0, m_1), $\mathcal{Z}^*$ activates a user with input $(\texttt{Sign}, m_0, v^*)$ and with $(\texttt{Sign}, m_1, v^*)$ and gets $\sigma(m_0)$ and $\sigma(m_1)$. $\mathcal{Z}^*$ sends $\{\sigma(m_0), \sigma(m_1)\}$ to $\mathcal{A}^*$. When $\mathcal{Z}^*$ gets $\tilde{b}$, the output of $\mathcal{A}^*$, $\mathcal{Z}^*$ outputs it. Consider the case that $\mathcal{Z}^*$ interacts with the ideal world. As shown in Figure 7.1, anyone cannot get the information about the linkage between the messages and documents. $\mathcal{A}^*$, thus, has no advantage in terms of the guess so that the probability of $\tilde{b} = 0$ is $\frac{1}{2}$. On the other hand, if $\mathcal{Z}^*$ interacts with the real world then the probability of $\tilde{b} = 0$ is $\frac{1}{2} + \varepsilon_1$. Therefore, $\mathcal{Z}^*$ can distinguish the two worlds with probability ε_1.

Next, consider the case that non-forgeability is broken. We can then construct $\mathcal{Z}^*$ that distinguishes the two worlds as in the above case. Let $\mathcal{A}^*$ be the adversary that breaks non-forgeability the probability ε_2. First, the activation of a signer yields v^*. $\mathcal{Z}^*$ then corrupts a user. $\mathcal{Z}^*$ simulates the interaction with Signer for $\mathcal{A}^*$ by delivering the outputs of $\mathcal{A}^*$ to corrupted User and vice versa. When $\mathcal{A}^*$ outputs a list $\{(m_1, \sigma(m_1)), \cdots, (m_j, \sigma(m_j))\}$, $\mathcal{Z}^*$ activates a party with $(\texttt{Verify}, m_i, \sigma(m_i), v^*)$ for all $1 \leq i \leq j$. If all results are $(\texttt{Accepted})$, $\mathcal{Z}^*$ outputs 1 and otherwise outputs 0. If $\mathcal{Z}^*$ interacts with the ideal world then for a forged pair the result is always $(\texttt{Reject})$. Thus the probability that $\mathcal{Z}^*$ outputs 1 is always 0. On the other hand, if $\mathcal{Z}^*$ interacts the real world then the probability that $\mathcal{Z}^*$ outputs 1 is ε_2. Therefore, $\mathcal{Z}^*$ can distinguishes with ε_2. □

Lemma 2 and Lemma 3 lead Theorem 1.

8.2 UC-secure blind signatures are not equivalent to SB-secure blind signatures

In this chapter we show;

Theorem 4. *The class of UC-secure blind signatures is a proper subset of that of SB-secure blind signatures for static adversaries, assuming one-way permutations.*

To prove Theorem 4 we show that there exists a blind signature protocol that is SB-secure but not UC-secure. At first consider the following protocol π_1. We describe the overview of π_1 in Figure 8.1. π_1 is a two-party protocol and, as input, a user is given m and a signer is given 1^k where k is a security parameter. Before starting the computations, the signer runs key algorithm $Gen(1^k)$, obtains (pk, sk) and sends pk to the user. At the end of the computations, π_1 outputs pk to both parties. Additionally it outputs $\sigma(m)$ to the user. The user learns nothing about the signer's input and the signer learns nothing about m. We can actually construct such a protocol by using the two-party completeness theorem (see chapter 8.1) [12]. We show that π_1 is SB-secure but not UC-secure to prove Theorem 4.

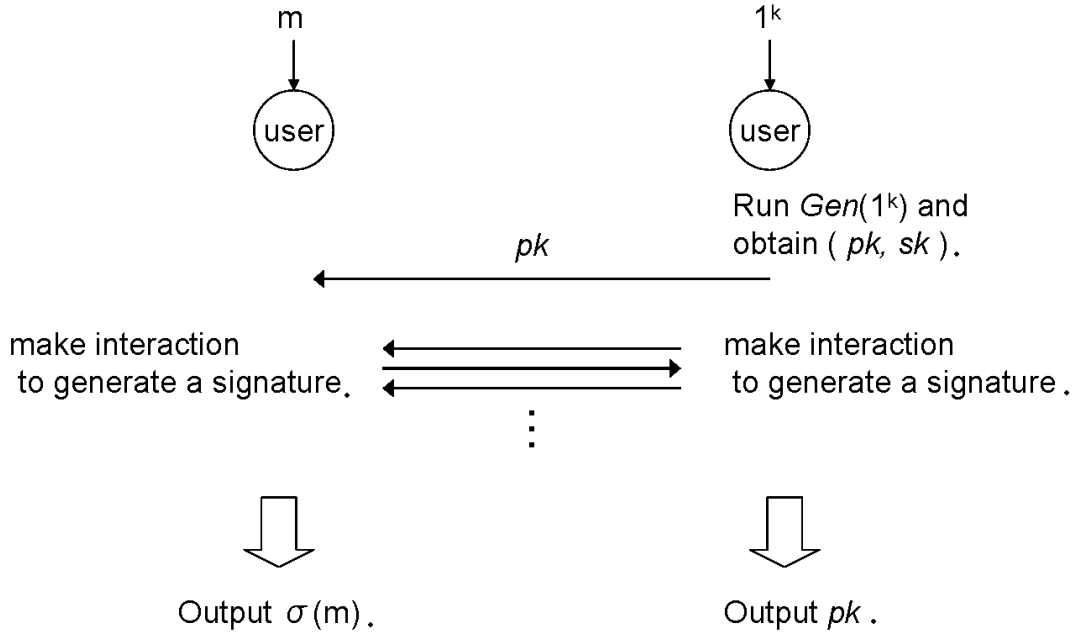

Figure 8.1: Protocol π_1

First, we show;

Claim 1. *π_1 is SB-secure.*

Proof: We show if π_1 is not SB-secure then the two-party completeness theorem is broken. First, consider the case that strong blindness is broken. This case implies that a malicious signer can get some information about m, which is clearly contradicts the property of zero-knowledge proofs. Recall that the two-party completeness theorem uses zero-knowledge proofs to verify that parties proceeds according to the protocol. Next, consider the case that non-forgeability is broken. There exist two cases that non-forgeability is broken. First, a pair randomly generated passes the verification with non-negligible probability. Second is the user is able to generate fake pairs by using secret data of the signer. We can ignore the first case so that we consider the second case. It , however, contradicts the two-party completeness theorem that no party can get any information except the output of the protocol. This completes the proof. □

Next, we show;

Lemma 4. *π_1 is not UC-secure.*

Proof: We show that π_1 satisfies the condition under that no two-party protocol can be UC-secure. Canetti, Kushilevitz and Lindell defined *unpredictability* and proved that an unpredictable probabilistic two-party function f cannot be UC-realized [6]. Unpredictability means that the output of some protocol does not depend on the inputs of both parties. Let $f_k : X \times X \rightarrow \{0,1\}^*$ be a probabilistic function that is parameterized by k. They showed the following two definitions;

Definition 3. $x_1(x_2) \in X$ is said a $P_1(P_2)$-safe value for $p(\cdot)$ and k if for every $x_2(x_1) \in X$ and all possible output values $v \in \{0,1\}^*$ it holds that $\Pr[f_k(x_1, x_2) \neq v] > \frac{1}{p(k)}$

Definition 4. $f = f_k$ is unpredictable if there exists a polynomial $p(\cdot)$ such that for infinity $k's$, there exist P_1-safe values and P_2-safe values for $p(\cdot)$ and k

Safe values have the property that they include non-trivial distributions over the output, that is, if P_1 inputs a safe value, the output of the function with

security parameter k is chosen from a non-trivial distribution regardless of the P_2's input.

Now, the output of π_1 includes pk and it is generated by Gen which is a probabilistic algorithm so that π_1 is a probabilistic protocol. We consider f_k, P_1, x_1, P_2, x_2 and v as π_1, a user, m, a signer, 1^k and $(pk, \sigma(m))$ respectively. If a security parameter k is specified 1^k is fixed. First, we show that there exists P_1-safe value for $p(k) = k^c$ (where, c is a constant). Assume that there exists no P_1-safe value, that is for any m there exist $1^{\hat{k}}$ and $(\hat{\sigma}, \hat{pk})$ such that $\Pr[\pi_1(m, 1^{\hat{k}}) \neq (\hat{\sigma}, \hat{pk})] < \frac{1}{p(k)}$. Now,

$$\begin{aligned} & \Pr[\pi_1(m, 1^{\hat{k}}) \neq (\hat{\sigma}, \hat{pk})] < \frac{1}{p(k)} \\ \Leftrightarrow \quad & \Pr[\pi_1(m, 1^{\hat{k}}) = (\hat{\sigma}, \hat{pk})] > 1 - \frac{1}{p(k)} \end{aligned} \tag{8.1}$$

Here, we take the strong blindness experiment. After choosing (m_0, m_1) $\mathcal{A}$ randomly chooses k^*. $\mathcal{A}$ then simulates $\pi_1(m_0, 1^{k^*})$ and gets $(\sigma^*(m_0), pk^*)$. $\mathcal{A}$ outputs (pk^*, m_0, m_1). When he is given (σ_b, σ_{1-b}), if $k^* = \hat{k}$, that is, $(k^*, \sigma^*(m_0)) = (\hat{k}, \hat{\sigma})$, then, from the expression (8.1), $\mathcal{A}$ succeeds the guess with over $1 - \frac{1}{p(k)}$. Because k is chosen uniformly, the probability of $k^* = \hat{k}$ is $\frac{1}{k}$. Thus, the probability that $\mathcal{A}$ succeeds the guess is greater than $\frac{1}{k}\left(1 - \frac{1}{p(k)}\right)$, which contradicts to strong blindness.

Next, we show that there exist P_2-safe value. As the same with above, we assume that 1^k is not a P_2-safe value, that is, for any 1^k there exist $\hat{m}$ and $(\hat{\sigma}, \hat{pk})$ such that

$$\Pr[\pi_1(\hat{m}, 1^k) = (\hat{\sigma}, \hat{pk})] > 1 - \frac{1}{p(k)}. \tag{8.2}$$

Here, we take non-forgeability experiment. $\mathcal{A}$ then randomly chooses k^* and locally runs $Gen(1^{k^*})$ and gets (pk', sk'). $\mathcal{A}$ randomly generates m^*, simulates the behavior of the signer using (pk', sk') and gets $(m^*, \sigma(m^*))$. After one interaction with the signer $\mathcal{A}$ gets $\{(m_0, \sigma(m_0))\}$, where $\pi_1(m_0, 1^{k_0}) = (\sigma(m_0), pk_0)$. Now, pk_0 is output by $Gen(1^k)$ regardless of m, thus, from the expression (8.2), the probability of $pk_0 = \hat{pk}$, that is, $(\sigma(m_0), pk_0) = (\hat{\sigma}, \hat{pk})$ is greater than $1 - \frac{1}{p(k)}$. Thus, if $k^* = k_0$ the probability of $pk' = \hat{pk}$, that is, $(\sigma(m^*), pk') =$

$(\hat{\sigma}, \hat{pk})$ is greater than $1-\frac{1}{p(k)}$. Here, $Verify(m_0, \sigma(m_0), pk_0)$ surely passes. The probability that $Verify(m^*, \sigma(m^*), pk')$, is greater than $\left(1-\frac{1}{p(k)}\right)^2$. Because k^* is randomly chosen the probability of $k^* = k_0$ is $\frac{1}{k}$. Thus, the probability that $(m^*, \sigma(m^*))$ passes the verification is greater than $\frac{1}{k}\left(1-\frac{1}{p(k)}\right)^2$, which contradicts to non-forgeability.

Therefore, π_1 is unpredictable so that we can say that π_1 is not UC-secure. This completes the proof. □

Next, we show;

Lemma 5. *If a blind signature protocol is UC-secure then it is also SB-secure*

Proof: As in the proof of Lemma 3, we show that if there exist a adversary that breaks SB-security we can construct an environment that distinguishes the two worlds. Let $\mathcal{A}^*$ be an adversary that breaks SB-security with ε. We then construct $\mathcal{Z}^*$ that distinguishes the two worlds. Because non-forgeability is the same in Definition 1 and in Definition 2, if $\mathcal{A}^*$ breaks non-forgeability we can construct $\mathcal{Z}^*$ that distinguishes the two worlds as described in Lemma 3. Assume that $\mathcal{A}^*$ breaks strong blindness. We then construct $\mathcal{Z}^*$ as follows. $\mathcal{Z}^*$ runs $\mathcal{A}^*$ locally in itself. First, $\mathcal{Z}^*$ corrupts a signer and receives the $\mathcal{A}^*$'s output. When $\mathcal{A}^*$ outputs $\{m_0, m_1, pk^*\}$ then $\mathcal{Z}^*$ activates a user with m_0 and m_1. In each interaction, $\mathcal{Z}^*$ sends a pk^* as a verification key. $\mathcal{Z}^*$ deliver messages from the user to $\mathcal{A}^*$ and vice versa during each interaction. When the user outputs two signatures $\sigma(m_0)$ and $\sigma(m_1)$, $\mathcal{Z}^*$ sends $\{\sigma(m_0), \sigma(m_1)\}$ to $\mathcal{A}^*$ in this order. When $\mathcal{A}^*$ outputs $\tilde{b}$, $\mathcal{Z}^*$ outputs it. If $\mathcal{Z}^*$ interacts with the ideal world then $\mathcal{A}^*$ has no advantage in terms of the guess so that the probability of $\tilde{b} = 0$ is $\frac{1}{2}$. On the other hand, if $\mathcal{Z}^*$ interacts with the real world then the probability of $\tilde{b} = 0$ is $\frac{1}{2} + \varepsilon$. Therefore, $\mathcal{Z}^*$ can distinguish the two worlds with probability ε. □

Lemma 4 and Lemma 5 lead Theorem 4.

As remark, we showed that the UC security is not equivalent to two other securities by the two difference ways. In the proof of Theorem 4, we introduce an instance of the protocol and showed that the instance is secure but not UC-

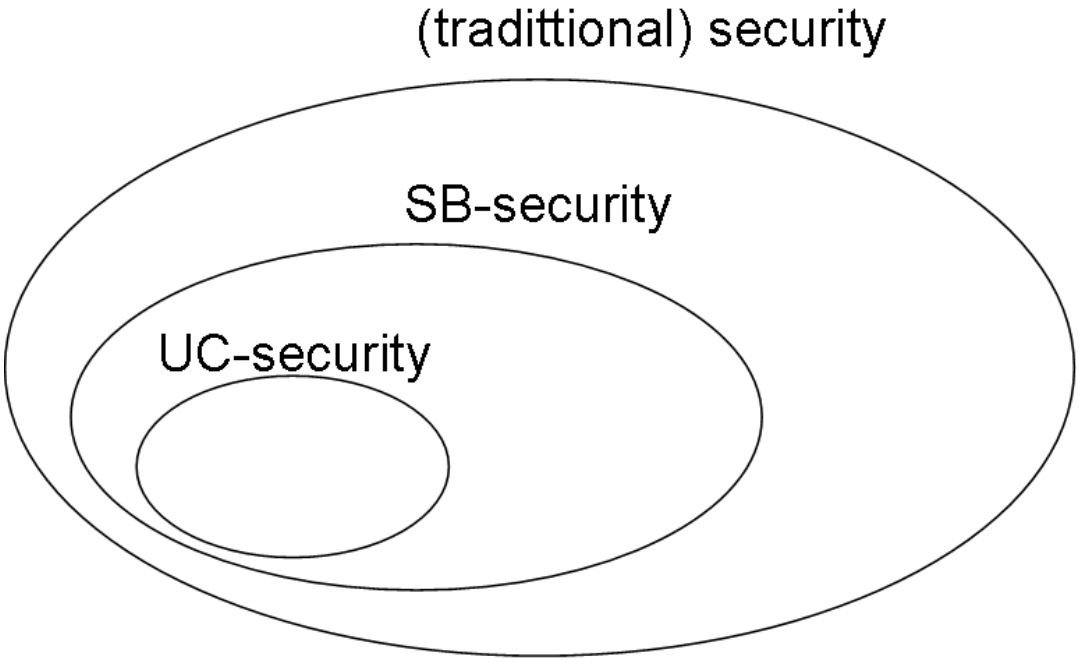

Figure 8.2: Relation of the security of blind signatures

secure. On the other hand, In the proof of Theorem 5, we use the theorem for two-party protocols and show that blind signature protocols are generally UC-secure even if they are SB-secure in the plain model.

References

[1] M. Abe, 'Universally Verifiable mix-net with Verification Work Independent of the Number of Mix-centers', Eurocrypt 98, pp. 437-447, 1998.

[2] Manuel Blum, 'Coin flipping by telephone', IEEE Spring COMPCOM, pp.133-137, Feb. 1982.

[3] J Cohen Benaloh and D Tuinstra, 'Receipt-Free Secret-Ballot Elections', STOC 94, pp.544-553, 1994.

[4] Ran Canetti, 'Universally Composable Security: A new paradigm for Cryptographic Protocols', 42nd FOCS, 2001. Full version available at *http://eprint.iacr.org/2000/067/*.

[5] Ran Canetti and Marc Fischlin, 'Universally Composable Commitments', Proceedings of CRYPTO 2001.

[6] Ran Canetti, Eyal Kushilevitz and Yeheuda Lindell, 'On the Limitations of Universally Composable Two-Party Computation Without Set-up Assumptions', Proceedings of EUROCRYPT 2003.

[7] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, Amit Sahai, 'Universally Composable Two-Party and Multi-Party Secure Composition', Proceedings of STOC 02.

[8] David Chaum, 'Blind Signatures for Untraceable Payments', Proceedings of CRYPTO 82.

[9] David Chaum, 'Untappable Electronic Mail, Return Addresses and Digital Psuedonyms', Communication of ACM, CACM 81, vol. 24, No. 2, pp. 84-88, 1981.

[10] ElGamal T, 'A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms', IEEE Trans. on Information Theory, IT-31, 4, pp.469-472, 1985.

[11] Oded Goldreich, Sivio Micali and Avi Wigderson, 'Proofs that Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems', Journal of ACM, Vol. 38, No. 1, pp. 691–729, 1991.

[12] Oded Goldreich, Sivio Micali and Avi Wigderson, 'How to Play Any Mental Game', Proceedings of STOC 87.

[13] M. Jakobsson, A. Juels, 'Millimix: Mixing in small batches', DIMACS Techical report 99-33, June 1999.

[14] Ari Juels, Michael Luby and Rafail Ostrovsky, 'Security of Blind Digital Signatures', Proceedings of CRYPTO 97.

[15] Leslie Lamport, 'Constructing digital signatures from one-way functions', SRI intl. CSL-98, October 1979.

[16] Moni Naor and Moti Yung, 'Universal One-Way Hash Functions and Their Cryptographic Applications', Proceedings of STOC 89.

[17] Tatsuaki Okamoto, 'An Efficient Divisible Electronic Cash Scheme', Proc. of Crypt 95, LNCS 963, pp.438-451, 1995.

[18] Tatsuaki Okamoto, 'Receipt-Free Electronic Voting Schemes for Large Scale Elections', Proc. of Crypt 95, LNCS 963, pp.438-451, 1995.

[19] David Pointcheval and Jacques Stern, 'Security Arguments for Digital Signatures and Blind Signatures', Proceedings of EUROCRUPT 96.

[20] Kazue Sako, Joe Killian, 'Receipt-Free Mix-Type Voting Scheme -A practical solution to the implimentation of a voting booth-', LNCS 921, pp. 393-403, 1995.

[21] Y.Tamura ,T.Siotsuki, A.Miyaji, 'Efficient Proxy-bidding System', IEICE Trans., Vol. J87-A, No.6, 835-842, 2004.

[22] Douglas Wikström, 'A Universally Composable Mix-Net', TCC 2004, LNCS 2951, pp.317-335, 2004.

[23] Andrew Chi-Chih Yao, 'How to Generate and Exchange Secrets', Proceeding of 27th FOCS, 1986, pp.162-167.

Acknowledgments

The author would like express his thanks to all the teaching staffs in the Okamoto laboratory in Department of Social Informatics , and all advisors, Mizuho Iwaihara, Yusuke Yokota, Syoichi Hirose, and Keisi Tajima, for their contribution.