A Study on Provable Security of Signature Schemes for Multiple Signers

Graduate School of Systems and Information Engineering

University of Tsukuba

March ２０１４

Naoto Yanai

Abstract

In this paper, we research provably secure signature schemes with multiple signers. Signature scheme for multiple signers is an extension of digital signatures in which the signers are associated with a document and the validity of the document is guaranteed by all the signers. This primitive is an important primitive from viewpoints of basic cryptographic theory and applications. In this paper, we discuss signature schemes for multiple signers in the following terms: (1) we propose optimized schemes for applications, and (2) we consider mathematical properties toward a provably secure generic construction. The reason is as follows. First, generally speaking, security proof is a difficult task, and there are several papers whose proposed scheme is broken later. Hence, we shed light on the properties for proving security in order to make a proof easy. Meanwhile, applications in the real world are constructed under various specifications, and these specifications are different for each application. Thus, we also construct optimized schemes satisfying the specifications. More specifically, we focus on following schemes.

Optimized Construction We propose ordered multisignature schemes, a structured multisignature scheme, a BGP-aiding aggregate signature scheme and a certificateless aggregate signature scheme as optimized constructions.

Ordered multisignatures [30] are a variant of multisignatures where signers guarantee the signing order among a group of the signers in addition to an improvement of the efficiency. By adopting these signatures to routing protocols, we can overcome DDoS attacks or provide in-band network fault localization. We give the details in Chapter 3.

Structured signatures [19] are an extension of ordered multisignatures. Whereas the ordered multisignatures deal with only a sequential process, the structured signatures deal with both the sequential process and a parallel process of multiple signers. The structured signatures have been expected to extend the content-editing system. We give the details in Chapter 4.

BGP-aiding aggregate signatures are a variant of aggregate signatures where these are specified which is designed for secure-border gateway protocol (S-BGP) [52]. Although aggregate signatures have been proposed for the BGP security, state-of-the-art schemes have a gap with a specification of the border gateway protocol [79]. We specify requirements of an aggregate signature scheme for the BGP security and call such a scheme BGP-aiding aggregate signature scheme. We also propose an instantiation of BGP-aiding aggregate signature scheme. We give the details in Chapter 5.

In Chapter 6, we focus on certificateless cryptosystem [2]. Certificateless cryptosystem is a hybrid primitive of the conventional public key infrastructure (PKI) and ID-based cryptosystem [86], and is a recent primitive. We discuss a signature scheme for multiple signers in the certificateless cryptosystem, and this primitive is suitable in ID federation in cloud service.

Toward Generic Construction We propose an unrestricted aggregate signature scheme as a general consideration of signature scheme for multiple signers. Existing signature schemes for multiple signers allow signers to sign only once. The reason of the restriction comes from security: more precisely, if the signers sign multiple times, then the security cannot be proven. Signature scheme allowing signers to sign multiple times is considered as the most generic construction of digital signature and is called unrestricted aggregate signatures. We discuss a construction of the unrestricted aggregate signatures and their property to prove security. We found a fact that all the schemes described above includes the properties and so these are provably secure. Namely, these schemes can be constructed from an unrestricted aggregate signature scheme. We give the details in Chapter 7.

Acknowledgement

I would like to express gratitude to my supervisors, Professor Eiji Okamoto and Professor Masahiro Mambo, for their guidance and encouragement during my studies. I have learned a lot, including my career option, from them and the attitude towards a research and education. I would also like to thank my advisors, Associate Professor Takashi Nishide, Associate Professor Jean-Luc Beuchat, Lector Akira Kanaoka and Assistant Professor Naoki Kanayama, for their invaluable comments in this thesis. I would also like to thank Associate Professor Kazuki Katagishi for his feedbacks in this thesis.

I would like to thank the present and past Okamoto-lab members, Mr. Hiroki Kunitake, Mr. Hayato Iseri, Mr. Teppei Yamazaki, Mr. Hirotaka Kokubo, Mr. Shun Kubota, Dr. Masayuki Okada, Mr. Tadahiko Itoh, Mr. Yusuke Niwa, Mr. Kazutaka Saito, Mr. Ichita Higurashi, Mr. Ijiro Hideaki, Mr. Moreno Alberto, Dr. Carine Yi, Dr. Kuniaki Tanaka, Mr. Masahiko Katoh, Mr. Yuki Higuchi, Mr. Nobuaki Kitajima, Mr. Kenta Ishii, Mr. Keita Koiwa, Mr. Shuai Chen, Mr. Yang Liu, Mr. Yukou Kobayashi, Mr. Satoshi Mitsui, Mr. Nasato Goto, Mr. Yu Liu, Mr. Kazuya Tanaka, Mr. Keisuke Hasegawa, and Mr. Kazuma Tanaka, for many interesting discussions on the topics related and not related to my research area. Especially, I would like thank Dr. Carine Yi, Mr. Nobuaki Kitajima and Mr. Yukou Kobayashi. They always took time for listening to my talks, including the research topics and many random chat. I would like to thank external Okamoto-lab visitors, Professor Xun Yi and Associate Professor Raylin Tso, who gave helpful suggestions and comments on this work.

I would like to thank the researchers at AIST, Dr. Goichiro Hanaoka, Dr. Koji Nuida, Dr. Nujttapong Attrapadung, Dr. Takahiro Matsuda and Dr. Tadanori Teruya, and members of Shin-Akarui-Angou-Benkyou-Kai, Dr. Keita Emura, Dr. Yoshikazu Hanatani, Mr. Michitomo Yamaguchi, Mr. Shota Yamada, Mr. Yusuke Sakai, Mr. Satsuya Ohata, Mr. Takashi Yamakawa, Mr. Kazuma Ohara, Ms. Ai Ishida, and Ms. Sanami Nakagawa. Especially, I would like to thank Dr. Goichiro Hanaoka and Dr. Koji Nuida

for their discussions, Dr. Takahiro Matsuda and Dr. Tadanori Teruya for their various interesting talks. I also would like to thank Associate Professor Eikoh Chida for his discussion. I also would lke to thank the researchers at Tokyo Science University, Mr. Masaki Inamura, at Mitsubishi, Dr. Yutaka Kawai, and at NTT Software, Mr. Satoru Kanno, for their interesting talks which were always helpful to me.

Finally, I would like to thank my mother and my grandma for their invaluable supports.

Contents

Chapter 1

Introduction

1.1 Overview

In this paper, we research provably secure signature schemes with multiple signers. In the recent electronic society, communication protocols such as SSL are crucial, and these protocols are constructed from cryptographic primitives such as public key cryptography. Provable security of a cryptographic scheme is to mathematically prove that an adversary have to solve some problem which is computationally infeasible to solve in order to break the scheme. In other words, we can analyze security of the scheme by understanding a security proof. Security of several cryptographic protocols such as ISO/IEC 9796-2 was not proven, and such schemes were broken even after these were standardized. By proving the security, cryptographic primitives are guaranteed to be secure even against any unknown attack. Provable security is crucial for modern cryptography in a theoretical and also practical sense. In such a framework, we focus on digital signature. Digital signature is a cryptographic framework to guarantee the validity of an electronic document, and each signer has a pair of a secret key and a public key. A signer signs the document by the secret key, and the validity of a signature is publicly verifiable by the public key. Verifiability of digital signatures supports all of elements for information security, i.e., confidentiality, integrity and availability. Integrity is to guarantee the validity of the data and corresponds to a well-known capability of a digital signature scheme. Confidentiality is provided by an encryption scheme in general, but can be also provided via an access control by user authentication. An authentication scheme is constructible from any digital signature scheme. For availability, we can utilize cloud computing, and cloud services are based

on the authentication scheme or availability of networks. Threats against the networks are denial-of-service (DoS) attack or route hijacking, but these can be overcome by digital signatures. Namely, the digital signatures are applicable in many situations. A signature scheme for multiple signers we discuss in this paper is a generic extension of a digital signature scheme to the multiuser setting, and is an important primitive from viewpoints of basic cryptographic theory and applications.

In this paper, we discuss signature schemes for multiple signers on the following terms: (1) we propose optimized schemes for applications, and (2) we consider mathematical properties toward a provably secure generic construction. The reason is as follows. First, generally speaking, security proof is a difficult task, and there are several papers whose proposed scheme is broken later. Hence, we shed light on the property for proving security in order to make a proof easy. Meanwhile, applications in the real world are constructed under various specifications, and these specifications are different for each application. Thus, we also construct optimized schemes satisfying the specifications.

1.2 Security Threats in Real World

There are many security threats in the real world. For instance, we describe several ones in the white paper by ENISA [22].

Distributed Denial of Service Denial of Service (DoS) attack is an attack where an attacker sends a large amount of packets to a server in order to bring down the server. Distributed Denial of Service (DDoS) is a variant of DoS attack where a large number of computers in multiple networks participate in this attack. A feature of DDoS attack is that the computers send the packets by controlling a malicious third party, and so owners of the computers cannot notice a fact that they participate in the attack. It is difficult for a system provider to distinguish access by the attack and access by a normal user.

Network Management In order to utilize services in the real world, network access is crucial. A malicious user can prevent users fro, utilizing any service by causing network failure such as an injection of a fake path. For instance, there is a case of YouTube hijacking [80].

Malicious Insider In general, a malicious provider can access any data stored in a server and so can potentially have an impact on the integrity. Security against such a malicious provider is a new security issue which has never been considered in a conventional environment.

Cooperation between Different Cloud Each service provider has a different domain and policy. Hence, in a large scale network, services may become unavailable due to different providers.

1.3 Signature Scheme for Multiple Signers

Signature scheme for multiple signers is a cryptographic primitive for efficiently operating digital signature. Its definition is that resulting signature is shorter than n individual signatures, and so the main motivation is for reducing the signature size and the verification cost. Currently, another motivation is also for an extension of its capability in addition to the improvement of the efficiency. The existing signature schemes for multiple signers are classified into following two schemes.

Multisignatures Multisignatures [48] allow n signers to sign a common data, and researches of signature schemes for multiple signers were started from this primitive. Currently, many researchers focus on aggregate signatures described below, and so there are few researches of standard multisignatures.

Aggregate Signatures Aggregate signatures [17] allow n signer to sign n individual documents and compress these signatures into a single short signature. The aggregate signatures have an advantage for allowing each signer to deal with different data and so are called generalized multisignatures. In recent years, aggregate signatures have been proposed as researches of signatures for multiple signers.

1.4 Goal of This Work

The goal of this work is following two constructions: (1) we propose optimized schemes for applications, and (2) we consider mathematical properties toward a provably secure generic construction. In both multisignatures and aggregate signatures, there is no definition such as including all the existing schemes, and an individual definition exist for each capability. In addition,

these schemes are applied to devices with a low computational power, and so an optimized construction for each application is crucial. In particular, for the first goal, we construct schemes sufficient for overcoming the problems described above. Meanwhile, there is no scheme including all the existing schemes, because the multiuser setting is a general extension of a conventional digital signature scheme and includes many possible capabilities. Thus, for the second goal, we construct a more generalized scheme for supporting the other schemes by shedding light upon the properties for the security proof. More specifically, we focus on following schemes.

1.4.1 Optimized Construction

Ordered Multisignatures Ordered multisignatures [30] are a variant of multisignatures where signers guarantee the signing order among a group of the signers in addition to an improvement of the efficiency. By adopting these signatures to routing protocols, we can overcome DDoS attack or provide in-band network fault localization. We give the details in Chapter 3.

Structured Multisignatures Structured signatures [19] are an extension of ordered multisignatures. Whereas the ordered multisignatures deal with only a sequential process, the structured signatures deal with both the sequential process and a parallel process of multiple signers. The structured signatures have been expected to extend the content-editing system. We give the details in Chapter 4.

BGP-Aiding Aggregate Signatures Aggregate signatures have been proposed for secure-border gateway protocol (S-BGP) [52]. However, state-of-the-art schemes have a gap with a specification of the border gateway protocol [79]. We propose an aggregate signature scheme which is designed for BGP and called the scheme as BGP-aiding aggregate signature scheme. We give the details in Chapter 5.

Certificateless Aggregate Signatures Certificateless cryptosystem [2] is a hybrid primitive of the conventional public key infrastructure (PKI) and ID-based cryptosystem [86], and is a recent primitive. In this paper, we also discuss a signature scheme for multiple signers in the certificateless cryptosystem, and this primitive is suitable in ID federation in cloud service. We give the details in Chapter 6.

1.4.2 Toward Generic Construction

Unrestricted Aggregate Signatures Existing signature schemes for multiple signers allows signers to sign only once. The reason of the restriction is for security: more precisely, if the signers sign multiple times, then the security cannot be proven. Signature scheme allowing signers to sign multiple times is considered as a most generic construction of digital signature and is called unrestricted aggregate signatures. We discuss a construction of the unrestricted aggregate signatures and their properties to prove security. Through constructing the schemes described above, we found a fact that these schemes include the properties and so these security are provable. Namely, these schemes can be constructed from an unrestricted aggregate signature scheme. We give the details in Chapter 7.

1.4.3 Paper Organization

In the rest parts of this paper, we describe some knowledges to understand this work in Chapter 2. In Chapter 3, we discuss an ordered multisignature scheme. In Chapter 4, we discuss a structured multisignature scheme. We propose a BGP-aiding aggregate signature scheme in Chapter 5, and then propose a certificateless aggregate signature scheme In Chapter 7, we describe an unrestricted aggregate signature scheme. Finally, in Chapter 8, we describe conclusion.

Chapter 2

Preliminaries

2.1 Mathematical Notions

2.1.1 Notation

We introduce the notation used in this paper. Let n denote the number of signers. We denote by m a message to be signed, by σ_i a signature generated by the ith signer, by pk_i a public key of the ith signer, by sk_i a secret key of the ith signer and by $\perp$ an error symbol. We denote by $|x|$ the size of any set x, by $a \parallel b$ a concatenation of elements a and b, from which a and b can be recovered.

2.1.2 Group

A group is a set $\mathbb{G}$ along with a binary operation $*$ for which the following conditions hold:

- (Closure) For all $g, h \in \mathbb{G}$, $g * h \in \mathbb{G}$.
- (Associativity) For all $g_1, g_2, g_3 \in \mathbb{G}$, $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$.
- (Existence of an Identity) There exists an identity $e \in \mathbb{G}$ such that for all $g \in \mathbb{G}$, $g * e = e * g = g$.
- (Existence of Inverse) For all $g \in \mathbb{G}$ there exists an element $h \in \mathbb{G}$ such that $g * h = h * g = e$. Such an h is called an inverse of g.

We denote by $(\mathbb{G}, *)$ a group $\mathbb{G}$ along with a binary operation $*$. $(\mathbb{G}, *)$ is a commutative group if, for all $g, h \in \mathbb{G}$, $g * h = h * g$ holds. Otherwise, we say $(\mathbb{G}, *)$ is a non-commutative group. When $\mathbb{G}$ has a finite number

of elements, we say $\mathbb{G}$ is a finite group and let $\#\mathbb{G}$ denote the order of the group. Here, we say that the order of $g \in \mathbb{G}$ is the smallest positive integer i with $g^i = 1$. If $\mathbb{G}$ is a group, a set $\mathbb{G} \subseteq \mathbb{G}$ is a subgroup of $\mathbb{G}$ if $\mathbb{H}$ itself forms a group under the same operation associated with $\mathbb{G}$. If there exists an element $g \in \mathbb{G}$ that has order a where a is the order of $\mathbb{G}$, then the subgroup $\mathbb{G}' = \{g^1, g^2, \cdots, g^a\}$ generated by g is identical to $\mathbb{G}$. In this case, we call $\mathbb{G}$ a cyclic group and say that g is a generator of $\mathbb{G}$.

2.1.3 Ring

A ring is a set $\mathbb{R}$ along with binary operation $+, \cdot$ for which the following conditions hold:

- $\mathbb{R}$ is a commutative group with $+$.
- (Closure) For all $g, h \in \mathbb{R}$, $g \cdot h \in \mathbb{R}$.
- (Associativity) For all $g_1, g_2, g_3 \in \mathbb{R}$, $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.
- (Distributivity) For all $g_1, g_2, g_3 \in \mathbb{R}$, $g_1 \cdot (g_2 + g_3) = g_1 \cdot g_2 + g_1 \cdot g_3$.

We denote by $(\mathbb{R}, +, \cdot)$ a ring $\mathbb{G}$ with operations $+$ and $\cdot$. $(\mathbb{R}, +, \cdot)$ is a commutative ring if, for all $g, h \in \mathbb{G}$, $g \cdot h = h \cdot g$ holds. Otherwise, we say $(\mathbb{R}, +, \cdot)$ is a non-commutative ring.

2.1.4 Homomorphism

2.1.5 Definition of Group Homomorphism

Let $(\mathbb{G}_1, \circ)$ and $(\mathbb{G}_2, \bullet)$ be groups. A (group) homomorphism is a function $f : \mathbb{G}_1 \to \mathbb{G}_2$ from $\mathbb{G}_1$ to $\mathbb{G}_2$ for which the following condition hold:

$$\forall a, b \in \mathbb{G}_1,\ f(a \circ b) = f(a) \bullet f(b).$$

2.1.6 Definition of Ring Homomorphism

Let $(\mathbb{R}_1, +, *)$ and $(\mathbb{R}_2, \dagger, \circ)$ be rings. A (ring) homomorphism is a function $f : \mathbb{R}_1 \to \mathbb{R}_2$ from $\mathbb{R}_1$ to $\mathbb{R}_2$ for which the following condition hold:

$$\forall a, b \in \mathbb{R}_1,\ f(a + b) = f(a) \dagger f(b),\ f(a * b) = f(a) \circ f(b)$$

2.2 Series-Parallel Graph

In this section we recall the definition of a series-parallel graph [12]. In Chapter 4 and Chapter 5, we utilize the series-parallel graph to represent a structure of signers. The definition is defined as follows:

Definition of Series-Parallel Graph

Let $\mathcal{G}$ be a set of graphs. A series-parallel graph is a graph generated by recursively applying either a serial graph or a parallel graph in an arbitrary order. More specifically, a series-parallel graph $G(I,T)$, which starts at the initial vertex I and terminates at the terminal vertex T, is defined as follows:

$G(I,T)$ is generated either by the following step 1 or step 2.

1. With a unique label i in $\mathcal{G}$, $G_i(I_i,T_i)$ is composed of one edge connecting I_i and T_i. We call such a graph an atomic graph and denote it by $\phi_i \in \mathcal{G}$.

2. For the step 2, either the following step (a) or step (b) is executed.

   (a) (Parallel Graph) Given n graphs $G_i(I_i,T_i)$ for $1 \leq i \leq n$, construct $G(I,T)$ by setting $I = I_1 = I_2 = \cdots = I_n$ and $T = T_1 = T_2 = \cdots = T_n$.

   (b) (Serial Graph) Given n graphs $G_i(I_i,T_i)$ for $1 \leq i \leq n$, construct $G(I,T)$ by setting $I = I_1, T_1 = I_2, \cdots, T_{n-1} = I_n$ and $T_n = T$.

   Intuitively, in the above definitions, constructing $G(I,T)$ means compositions of n atomic graphs $\phi_i \in \mathcal{G}$ for $i = [1,n]$ either as a serial one or a parallel one.

Composition of Graphs

For two graphs $\phi_1, \phi_2 \in \mathcal{G}$, we define a composition of parallel graphs as $\phi_1 \cup \phi_2$ and the composition of serial graphs as $\phi_1 \cap \phi_2$. In other words, $\phi_1 \cup \phi_2$ means to construct $G(I,T)$ by setting $I = I_1 = I_2$ and $T = T_1 = I_2$, and $\phi_1 \cap \phi_2$ means to construct $G(I,T)$ by setting $I = I_1, T_1 = I_2$ and $T_2 = T$. We denote by $\mathcal{T}(i)$ a set of graphs connecting to the initial vertex I_i of ith graph in a way such that $\mathcal{T}(i) = \{x \mid I_i = T_x \wedge 1 \leq x < i \wedge G_x(I_x,T_x) \subset \psi_n\}$, by $\mathcal{I}(i)$ a set of graphs connecting to the terminal vertex T_i of ith graph in a way such that $\mathcal{I}(i) = \{x \mid T_i = I_x \wedge i < x \leq n \ \wedge G_x(I_x,T_x) \subset \psi_n\}$, by $\{a_j\}_{j \in \mathcal{T}(i)}$, for all a, all a_j for $j \in \mathcal{T}(i)$. In other words, for all graphs, the composition of a graph ϕ_i and ψ_j for $j \in \mathcal{T}(i)$ can be denoted by $\psi_i := \phi_i \cap \left(\bigcup_{j \in \mathcal{T}(i)} \psi_j\right) = \phi_i \cap \psi_{\mathcal{T}(i)}$ where $\bigcup_x$ means iterations of the operation $\cup$ for all x. Similarly, $\bigcap_x$ can be defined as iteration of $\cap$.

Weight of Graph

We define a weight function $\omega_i(\psi_n)$ that represents a weight of each label i for a graph ψ_n. Intuitively, $\omega_i(\psi_n)$ means the number of paths including an edge with a label i from I_i to T_n for ψ_n. From Lemma 2.3 in [90], the weight of a graph consisting of n edges is $\omega_i(\psi_n) \leq 3^{\#\psi_n/3}$ for any graph ψ_n, where $\#\psi_n$ means the number of the edges in ψ_n.

2.3 Bilinear Maps

Definition 1 (bilinear maps)**.** Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of the same prime order p, and g be a generator of $\mathbb{G}$. A map $\boldsymbol{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is called a bilinear map if the following conditions hold: $\mathbb{G}$ and $\mathbb{G}_T$ are groups of the same prime order p and g is a generator of $\mathbb{G}$:

Bilinearity For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$, $\boldsymbol{e}(u^a, v^b) = \boldsymbol{e}(u, v)^{ab}$;

Non-degeneracy We have $\boldsymbol{e}(g, g) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity element of $\mathbb{G}_T$;

Efficient Computability There is an efficient algorithm to compute $\boldsymbol{e}(u, v)$ for any $u, v \in \mathbb{G}$.

Such a group $\mathbb{G}$ equipped with a bilinear map is called a bilinear group. In this paper, we assume that the discrete logarithm problem (DLP) for bilinear groups is hard. We call $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ a *pairing parameter*.

2.4 Security Assumption

2.4.1 Discrete Logarithm Problem

Let p be a prime number. The discrete logarithm problem in a cyclic group $\mathbb{G}$ with given generator g is to compute $\log_g h$ given a random element $h \in \mathbb{G}$ as input. In this paper, we assume the discrete logarithm problem as one-way function. We say this problem as DLP for short. Formally, we define the problem as follows:

Definition 2 ((t, ϵ)-DLP assumption)**.** We define the discrete logarithm problem with a security parameter 1^k as the problem of computing a from a given (p, g, g^a) in $\mathbb{G}^2$ with uniformly random $ain\mathbb{Z}_|$. We say that the (t, ϵ)-DLP assumption holds in $\mathbb{G}$ if there is no probabilistic polynomial-time algorithm that can solve the DLP problem in $\mathbb{G}$ with probability greater than ϵ within execution time t.

2.4.2 Computational Diffie-Hellman Assumption

The computational Diffie-Hellman (CDH) assumption is defined as follows.

Definition 3 ((t, ϵ)-CDH assumption)**.** We define the computational Diffie-Hellman (CDH) problem in bilinear groups with a security parameter 1^k as the problem of computing g^{ab} from a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ and a given $(g, g^a, g^b) \in \mathbb{G}^3$ with uniformly random $a, b \in \mathbb{Z}_p$ as input. We say that the (t, ϵ)-CDH assumption holds in $\mathbb{G}$ if there is no probabilistic polynomial-time algorithm that can solve the CDH problem in $\mathbb{G}$ with probability greater than ϵ within execution time t.

2.4.3 Identity-based Sequential Aggregate Signature CDH (IBSAS-CDH) Assumption

We recall the definition of the ID-based sequential aggregate signature CDH (IBSAS-CDH) assumption [15]. This is an interactive assumption, and Boldyreva et al. proved the hardness of this problem in [15]. The assumption is defined as follows:

Definition 4 ((t, q, ϵ)-IBSAS-CDH Assumption)**.** We define the IBSAS-CDH problem with a security parameter 1^k as the problem of computing, from a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ and a given tuple $(g, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2})$ with uniformly random $(a_1, a_2, b_1, b_2 \in \mathbb{Z}_p)$ as input, a tuple of $(g^{rx} g^{a_1 b_1} g^{m a_2 b_2}, g^r, g^x)$ for randomly generated numbers $(r, x) \leftarrow \mathbb{Z}_p$ under the setting that an oracle $\mathcal{O}^{\mathrm{IBSAS-CDH}}_{g, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}}$ that takes $m \in \mathbb{Z}_p$ as input and returns $(g^{rx} g^{a_1 b_1} g^{m a_2 b_2}, g^r, g^x)$ for any $(r, x) \leftarrow \mathbb{Z}_p$ is available, where the element m involved in each query should be different from the element m involved in the final output. We say that the (t, q, ϵ)-IBSAS-CDH assumption holds if there is no probabilistic polynomial-time algorithm that can solve the IBSAS-CDH problem with probability greater than ϵ within execution time t. Here the probabilistic polynomial-time algorithm can generate at most q queries to the oracle.

2.5 Provable Security

Provable security is a method to analyze security of a cryptographic scheme. Provable security consists of two steps, i.e., a definition of security and a construction of a reduction algorithm. First, we formalize a notion that a cryptographic scheme is broken as a Turing machine, and then show that any difficult problem which is an unsolved problem in the current technology can

be solved by utilizing the Turing machine, which is an adversary. This notion is called *reduction* and an algorithm to solve the difficult problem is called reduction algorithm. Constructing such a reduction algorithm means that the unsolved problem can be solved, and so the start for constructing the reduction is a wrong statement. Namely, we can show that an assumption in which the adversary exists is wrong and nobody breaks the scheme.

2.6 Programmable Hash Function

We briefly recall the notion of programmable hash functions [40] without formal definition. A programmable hash function is a keyed group hash function that can behave in two indistinguishable ways, depending on how the key is generated. If the ordinary key generation algorithm is used, then the hash function fulfills its normal functionality. The alternative trapdoor key generation algorithm outputs a key that is indistinguishable from the one output by the ordinary algorithm. It furthermore generates an additional secret information that depends on user-specified generators g and h from a group $\mathbb{G}$. This trapdoor information makes an input possible to relate the output of the hash function H to g and h. In particular, for any input x, one obtains integers a_x and b_x such that the relation $H(x) = g^{a_x} h^{b_x} \in \mathbb{G}$ holds. The Waters hash function [95] used in the Waters signature scheme is an instantiation of programmable hash functions with multi-generators.

2.6.1 Waters Hash Function

The Waters hash function is an instantiation of the programmable hash functions, and this constructions was proposed by Waters [95]. The programmable hash functions described above has been proposed after the Waters hash function. The Waters hash function consists of $\ell + 1$ generators, where ℓ is the length of a message, and many CDH-based schemes have been proposed via the function. We briefly recall the construction of the Waters hash function.

Waters Hash Function: Let $\mathbb{G}$ be a group family. Choose $\ell + 1$ generators $(u', u'_1, \cdots, u'_\ell) \in \mathbb{G}^{\ell+1}$. Then, for all message $m = (m_1, \cdots, m_\ell) \in \{0,1\}^\ell$, compute $H(m) = u' \prod_{i=1}^{\ell} u_i^{m_i}$.

2.7 Conventional Public Key Cryptosystem vs. ID-based Cryptosystem

Public key cryptography is a cryptosystem which gives each user two keys, a secret key and a public key. The secret key is utilized to generate signatures while the public key is published to verify the signatures. These are random strings, and so we need a mechanism to clarify a relation between these keys and their owner. Currently, public key infrastructure (PKI) has been utilized as such a mechanism. In PKI, users are given certificates to guarantee the relation between the keys and the owners by a trusted third party called certification authority (CA). This mechanism has been implemented in a web browser, and we can utilize it without without an expertise. The main problem in PKI is the management cost for the certificates. The cost is expensive in that it involves certificate revocation, distribution and verification of public keys. This problem in PKI is called *certificate management problem*.

An approach to bring down the cost is ID-based cryptosystem [86]. ID-based cryptosystem allows users to utilize any string such as user's ID as public keys. In general, ID is unique information for each user and is publicly known. Thus, the cost to manage the certificates can be reduced. However, ID-based cryptosystem has an inherent problem, called *key escrow problem*, in which a key generation center (KGC) knows secret keys for all users in the system. This problem occurs because the secret keys of all the users are computed from KGC's master secret key and strings published by the users. This implies that the KGC must be trusted in ID-based cryptosystem. In other words, malicious KGC's can easily operate contents of the users and ID-based systems intrinsically contain such an insider's threat. In fact, we cannot always trust KGC's since malicious KGC's who do not honestly run the algorithm exist [56].

Conventional public key cryptosystem via PKI and ID-based cryptosystem have individual advantages respectively, and we do not know that which one is the better cryptosystem. We know only a fact that conventional public key cryptosystem provides the level-3 security described in the next section while ID-based cryptosystem provides the level-1 security. On the other hand, ID-based cryptosystem may provide more applications as described in Chapter 5, and so we have to choose these cryptosystems carefully.

2.8 Girault's Security Level for Cryptosystem

In a security notion in public key cryptosystems, Girault [38] defined three security levels for a trusted authority as follows:

level-1 KGC knows a secret key for any user, and can impersonate the user with the secret key without being detected.

level-2 KGC does not know a secret key for any user, but can impersonate the user with the secret key without being detected by generating a fake secret key.

level-3 KGC does not know a secret key for any user. In addition, KGC cannot impersonate the user with the secret key even by generating a fake secret key since its impersonation can be detected.

2.9 Multisignatures and Aggregate Signatures

Multisignatures are a tool which can efficiently deal with signatures generated by multiple signers. Suppose that the number of signers is n, the total size of signatures in a multisignature scheme is less than n times of the size in a single signature scheme used, while in a trivial scheme, it is just n times of the size in the single signature scheme. As a basic model of an attack against multisignatures, the adaptively chosen message and active insider attack [73, 67] has been defined. Shortly, an adversary can execute the following scenario. First, during the key-generation phase, the adversary can register colluding signers adaptively as insiders. Second, during the signing phase, the adversary can conspire with the valid signers, and can generate signing queries to honest signers adaptively. The goal of the adversary is to forge a multisignature including at least one target signer. The security of most multisignature schemes is proved against this attack.

Aggregate signatures are known as generalized multisignatures in which each signer among a signing group signs an individual document, whereas the multisignatures allow signers to sign a common document. Currently, the main approach of signatures scheme for multiple signers is for the aggregate signatures. The security of the aggregate signatures can be discussed similarly as that of the multisignatures.

2.9.1 Notions of Security Model

There are four meta models as security of multisignatures, i.e., certified key model [13], key registration model [81], key verification model [5], and plain

public key model [10]. These models provide different levels of security, and we briefly describe each model below.

The certified key model is a model assuming that each signer knows a secret key corresponding to an own public key. This model implicitly consider a situation in which an attack during the key-generation phase called rogue-key attack [67], and the security proof is easy relatively to other models. Although this model is the weakest model in the above models, the assumption that the signer knows the secret key can be realized by well-known protocols such as the Schnorr identification scheme [84]. Thus the security of the most schemes has been proven under the model.

The key registration model is a model forcing signers to execute an identification scheme with a trusted party when they generate a pair of keys in order to obtain certificates of the keys. Intuitively, this model includes an identification implicitly assumed in the certified key model, and overcome the rogue key attack. Thus this model provides a stronger security than the certified key model.

The key verification model is a model forcing signers to execute an identification scheme similarly as the key registration model but in this model the signers generate their certificates by themselves. This model essentially makes the assumption of the key registration model in that the trusted party is unnecessary, and so this model is a better one than the key registration model.

The plain public key model is a model assuming only that each signer has a public key. In this model, signers only utilize given public keys, and a fact that they generate the keys honestly is no longer guaranteed. The reason why the plain public key model was proposed is due to the current mechanism of PKI. Whereas the rogue key attack can be overcome by the identification protocols, the existing specifications of PKI do not support such a protocol according to the paper [10]. Thus, the authors of the paper [10] pointed out that security of the multisignatures should be proven extra protocols such as the identification scheme. The plain public key model provides the security under such a scenario and so this is the strongest model in signature scheme for multiple signers.

Chapter 3

Ordered Multisignatures

3.1 Introduction

3.1.1 Motivation

The current Internet design aims to provide high security for routing protocols without decreasing availability. A main approach to provide such a capability is to combine the routing protocols with cryptography. For instance, secure-border gateway protocol (S-BGP) [52] or border gateway protocol security (BGPSEC) [59] are inter-domain routing protocols with digital signatures where each autonomous system (AS) signs its own path information. These protocols can prevent route hijacking by a virtue of the digital signatures, and is currently under consideration for standardization by IETF. Whereas cryptographic tools are expected to drastically extend a capability of network systems, they increase costs of network systems. Indeed, there are 36,000 ASes in the Internet [25], and in such a large scale network the cryptographic tools often force a large amount of loads against AS routers. In fact, developments of the above technologies have been prevented due to ballooning memory of the routers [89]. In this work, we try these problems in the point of view for cryptography. Namely, our goal is to propose a provably secure cryptographic scheme, which is practical even for large scale networks in terms of storage of memory, bandwidth overhead and scalability.

Multisignatures [48] in which n signers sign a message and compress n individual signatures into a single short signature are expected as a cryptographic approach to overcome such an overloading problem, because this primitive is suitable for devices with low-computational power and small storage such as routers in a term of compressing the signatures. Among

the multisignatures, ordered multisignatures [30], where signers also guarantee the signing order in addition to the efficiency improvement of ordinary multisignatures [48], have been expected to provide more useful systems [14] such as the data-plane security of border gateway [32] protocol or a detection of network faults.

However, there is no practical scheme whose security is proven under the well-known assumption such as the CDH assumption without random oracles. While an important point for the routing applications is simply the efficiency, we should also take into account the security from the following two reasons. Firstly, Canetti et al. [20] showed that there are signature and encryption schemes that are provably secure in the random oracle model [11] but become insecure in an implementation of the random oracles. This result indicates that the security should be proven without random oracles, i.e., in the *standard model*. Secondly, strong cryptographic assumptions may be broken by developments of the computer science such the algorithm by Cheon [24], and hence the security of a cryptographic scheme should be proven under a weaker assumption. Although there are several schemes in such an approach, the existing schemes require a large memory and hence is undesirable for its underlying purpose.

In this paper, we propose efficient and secure ordered multisignature schemes. Our schemes are provably secure under the CDH assumption without the random oracles [20] and fairly practical.

3.1.2 Our Contribution

We propose two provably secure ordered multisignature schemes without aggregate signatures under only the standard assumptions, i.e., the CDH problem without the random oracles. One of the schemes consists of the Waters hash function [95], and another scheme consists of the chameleon hash function [53]. Both of our schemes are relatively more efficient than the existing schemes in the sense that the number of elements of public keys and the number of bilinear maps, whose cost is heavy, are independent of the security parameter. Although several parts of the computations depend on the number of signers, these values are quite lesser than the security parameter to be used, i.e., the number of the signers is at most 20 as described in Section 3.6 while the security parameter is set to be at least 160 for 80-bit security. Hence, the proposed schemes are efficient in a practical scenario.

Our main approach is to eliminate the use of aggregate signatures [17]. While any aggregate signature scheme gives rise to an ordered multisignature scheme [14], the existing aggregate signature schemes under the CDH

Table 3.1: Evaluation of the schemes: We denote by ℓ the message length, by $\mathcal{P}$ the computational cost of one bilinear map, by $\mathcal{E}$ the computational cost of one exponentiation, by $\mathcal{H}$ the computational cost of one map-to-point, by $\mathcal{L}(p)$ the binary length of p, by k the security parameter, by ROM the random oracle model, by AS aggregate signatures and by OMS ordered multisignatures. Typical values for these parameters are $\mathcal{L}(p) = 176$ on a symmetric pairing for 80-bit security, $\ell = 176$, $n = 20$. and, with Type A curve in PBC library [65] according to [47], the cost per one $\mathcal{P}$ is 2.2078 msec, the cost per one $\mathcal{E}$ is 2.5591 msec and the cost per one $\mathcal{H}$ is 5.8960 msec.

Schemes	Computational Cost for Verifier	Signature Size	Public Key	Public Parameter	Proof Model
ECDSA [93]	$2n\mathcal{E}$	$2n\mathcal{L}(p)$	$\mathcal{L}(p)$	$\mathcal{L}(p)$	-
BGOY07 [14]	$3\mathcal{P} + n\mathcal{E} + \mathcal{H}$	$2\mathcal{L}(p)$	$3\mathcal{L}(p)$	$\mathcal{L}(p)$	ROM
LOSSW06 [64]	$3\mathcal{P}$	$2\mathcal{L}(p)$	$(\ell+2)\mathcal{L}(p)$	$\mathcal{L}(p)$	Standard
AGH10 [1]	$(\ell+3)\mathcal{P} + 2\mathcal{E}$	$2\mathcal{L}(p) + k$	$\mathcal{L}(p)$	$(\ell+5)\mathcal{L}(p)$	Standard
Our WH Scheme	$4\mathcal{P} + n\mathcal{E}$	$3\mathcal{L}(p)$	$3\mathcal{L}(p)$	$(\ell+3)\mathcal{L}(p)$	Standard
Our CH Scheme	$4\mathcal{P} + (4+n)\mathcal{E}$	$3\mathcal{L}(p) + 2k$	$3\mathcal{L}(p)$	$6\mathcal{L}(p)$	Standard

assumption in the standard model are inefficient since either the number of the bilinear maps or the number of elements of public key depends on the message length. Namely, utilizing these schemes in routers is impractical. Hence, we construct ordered multisignature schemes from scratch without the aggregate signatures. Meanwhile, in this approach as described in Section 3.3, there is some technical problem [14] that random numbers for simulating the signing oracle cannot be controlled because there are relatively more signature components in the ordered multisignatures than that in ordinary multisignatures.

We found that the problem can be overcome by giving individual random numbers to each component of ordered multisignatures, where the size of these values is independent of the number of signers. In this methodology, each random number is not affected by another one, and the signing oracle can be simulated as long as either random number is a known value. Intuitively, we can consider in this methodology a security proof of the ordered multisignatures as the combination of security proofs of two individual sig-

nature schemes for a single signer. Under this observation, we propose two schemes, one with the Waters hash and the other with the chameleon hash. We give the detail of this idea in Section 3.3.

In both schemes, the signature size and the number of bilinear maps are independent of the number of signers. In the Waters-hash-based scheme, even though the size of the public parameter depends on the message length, this scheme is a stateless signature scheme where the signer does not need to maintain state. On the other hand, the chameleon-hash-based scheme is a stateful signature scheme where the signer must maintain state, but the size of the public parameter in this scheme is quite small. The former scheme is stateless and hence the security can be guaranteed via a more generic setting. The later scheme is stateful but the memory storage becomes smaller. The computational time of Waters-hash-based scheme is lesser than one-tenth of that of the AGH10 scheme, and the memory size of routers with the chameleon-hash-based scheme becomes one-percent in comparison with the LOSSW06 scheme.

3.1.3 Applications

Data Plane Security in BGP One of possible applications of ordered multisignatures is an improvement of S-BGP [52]. Feamster et al. [32] pointed out that an AS router should reject packets from invalid sources. They called such a new capability the data-plane security, and an advantage of the data-plane security is that packets are certainly forwarded along with the own path information.

According to Boldyreva et al. [14], ordered multisignatures seem to be suitable for the data-plane security: more specifically, in order to provide the data-plane security, ASes can sign the data packets by using an ordered multisignature scheme where the signing order represents the actual forwarding path and the packets themselves correspond to messages. Each packet is signed by egress routers of ASes forwarding it, and the egress routers insert their signatures and public key information into each data packet. Ingress routers receiving the packets forward only packets with valid ordered multisignatures that followed an authenticated path, where the routers check that the validity of the signing order of the signatures with their own path information. If the verification algorithm rejects the signatures, the ingress routers will eliminate the packets. This approach is expected to prevent several attacks such as Denial of Service (DoS) attack. In comparison with ordinary signatures such as ECDSA, by a virtue of an aggregation of the signatures, the overloads of the routers can be reduced by $\frac{1}{20n}$ where n is

the number of ASes.

In-Band Network Fault Localization Network faults may introduce packet losses or degradations of the performance, and they also negatively affect the network troubleshooting. To address this problem, Motiwala et al. [69] proposed a method detecting the faults, called in-band network fault localization. In the system, data packets include probe packets for recording the IP address of routers along the path. More specifically, probe packets consist of three parts, i.e., (1) Flow, (2)Incr and (3)Router ID, and each part counts (1) the sequence number for this packet, (2) the number of packets lost by a router and (3) the hash of the router that lost packets, respectively. These values are also given to each router, and a router will find a difference in the values in the packets and the one stored at the router if any packet is lost. Main technical problems in this system are security and performance. One of threats is that a router forges the values in the packets in order to avoid that its trustworthiness is downgraded. In order to get rid of such a situation, data packets should be signed by ordered multisignatures. In comparison with standard signatures, their performances are drastically remedied using ordered multisignatures within $\mathcal{O}(n)$ from $\mathcal{O}(n^2)$, where each router corresponds to a signer, probe packets correspond to messages to be signed.

3.1.4 Related Work

The first multisignature scheme was proposed by Itakura and Nakamura [48], and the security was formalized by Ohta and Okamoto [73] and Micali et al. [67]. After, four security models have been proposed, i.e., the certified key model [13], the plain public key model [10], the key registration model [81], and the key verification model [5]. Among these models, the certified key model is suitable for considering properties in the security proof since the other models force to discuss too much complicated security.

The first ordered multisignature scheme was proposed by Doi et al. [30], and many such schemes without aggregate signatures [14, 19, 43, 51, 61, 62, 68, 90, 100] have been proposed. An important result is the order flexibility in [68] where neither order of signers nor signers themselves need to be designated beforehand. Schemes which do not meet the order flexibility have a restriction on the number of signers, i.e., about 8 signers for 80-bit security as shown in [90]. Meanwhile, these schemes are based on the well-known assumptions such as the DLP or the CDH problem. However, as described in Section 7.1, there is no scheme without the random oracles. To the best of

our knowledge, in order to construct such an ordered multisignature scheme, there is an aggregate signature scheme in advance.

Meanwhile, an aggregate signature scheme, which is a generalized multisignature allowing each signer to sign an individual document, was proposed by Boneh et al. [17] while Mitomi and Miyaji [68] was individually proposed an older scheme with such a message flexibility. Aggregate signatures give rise to ordered multisignatures where each signer signs a concatenation of a common message and a list of signers, and several aggregate signature schemes in the standard model have been proposed [1, 55, 54, 64, 82, 85]. Among them, the schemes whose security is based on the well-known problem such as the CDH problem are only in [1, 64]. However, the number of computations of the bilinear maps in the AGH10 scheme [1] and the number of elements of the public key in the LOSSW06 scheme [64] depend on the security parameter, which are quite large. Thus, these schemes are inefficient.

As approaches for combinations of digital signatures and routing protocols, Kent [52], Zhao et al. [108], and Boldyreva and Lychev [16] have proposed the following results. Kent [52] proposed S-BGP, and BGPSEC [59] is a protocol such as a specification of this protocol. After, Zhao et al. [108] proposed a path-authentication protocol with aggregate signatures as a variant of S-BGP. Boldyreva and Lychev [16] formalized several threats of the routing protocols and showed the relation between the unforgeability of digital signatures and the security.

3.2 Definition of Ordered Multisignatures

In this section, we define a syntax of an ordered multisignature scheme and the security. Ordered multisignatures are a natural extension of multisignatures where the signatures guarantee not only messages but also the signing order, and its signing procedure is not interactive but sequential among a group of signers. In other words, each signer signs any common message one by one and a case that multiple signers sign in parallel is out of the scope of this primitive.

3.2.1 The Syntax

An ordered multisignature scheme consists of the following algorithms. In this chapter, we define $\psi_i := pk_1 \parallel \cdots \parallel pk_i$ to be the signing order from the first signer to ith signer where $\psi_0 := \emptyset$, and denote by $|\psi_i|$ the number of signers in ψ_i.

Setup Given security parameter 1^k, generate a public parameter $para$.

Key Generation Given $para$, generate a secret key sk_i and its corresponding public key pk_i.

Signing Given a secret key sk_i, a public key pk_i, a message m, a multisignature σ' from the previous signers, and a signing order ψ_{i-1}, generate a signature σ. In any problem occur, output an error symbol $\perp$. Finally, set $\psi_i = \psi_{i-1} \parallel pk_i$ and output the signature σ on m in ψ_i, where for $i = 1$ set $\psi_1 = pk_i$ as an initial value.

Verification Given m, σ, ψ_n and $\{pk_i\}_{i=1}^n$, output *accept* or *reject*.

The correctness of the ordered multisignature scheme is defined as follows. In an ordered multisignature scheme, we say that the scheme is correct if, for all $para, sk_i$ and pk_i given by **Setup** and **Key Generation**, **Verification**$(m, \textbf{Signing}(sk_i, pk_i, m, \sigma', \psi_{i-1}), \psi_i, \{pk_j\}_{j=1}^i)$ outputs *accept*.

3.2.2 Security Model

In this model, there exist an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Our model is a variant of the certified key model [13]. The certified key model assumes that each user knows a secret key corresponding to its own public key. Although there are models providing stronger security such that there is no such an assumption, these models constrict understanding properties to prove the security due to the complicated analysis. Meanwhile, in majority of the existing schemes, the public key corresponding to a secret key x is g^x, and a knowledge of the secret key is easily guaranteed by the Schnorr identification scheme [84]. Hence, PKI-based security based on the certified key model is realistic.

$\mathcal{C}$ has a list $\mathcal{L}$ of certified keys that is used to certify users' own public keys. $\mathcal{A}$ can know all secret keys corresponding to public keys included in $\mathcal{L}$ except for the one given by $\mathcal{C}$ to a target signer. $\mathcal{A}$'s advantage is equal to the probability that $\mathcal{C}$ outputs *accept* in the following game. Similarly as the existing model of ordered multisignatures [14], our security model guarantees authenticity of the message signed by an honest signer and its position i in a path, but not which signers signed before or will sign after the i-th signer. We do not consider switching of the positions among colluding malicious signers. For instance as shown in [14], there are malicious signers corresponding to pk_1 and pk_3 colluding each other against an honest signer

corresponding to pk_2. Signers corresponding to pk_1 and pk_3 may be able to compute some signature σ on m in $\psi_2 = pk_1 \parallel pk_2$ after obtaining σ^* on m in $\psi_3^* = (pk_3 \parallel pk_2) \parallel pk_1$. According to [14], this setting seems to be acceptable in the application described in Section 7.1.3. Hereinafter, we denote by $x^{(i)}$ the value of the ith query for all x. $\mathcal{C}$ interacts with $\mathcal{A}$ as follows:

Initial Phase Generate a public parameter $para$ by **Setup** and a a pair of challenger keys (sk^*, pk^*) by using **Key Generation**. Then, register pk^* in $\mathcal{L}$, and run $\mathcal{A}$ with $para$ and pk^* as input.

Certification Query Given (sk_i, pk_i) by $\mathcal{A}$ check that sk_i is a secret key corresponding to pk_i, and then register pk_i in $\mathcal{L}$ if so. Otherwise, return $\perp$.

Signing Query Given a signing query $(m^{(h)}, \sigma', pk^*, \psi_{i^{(h)}-1})$ by $\mathcal{A}$ of the target signer pk^* for all $i \in [1, n]$, check that the following conditions hold for the query: **Verification** algorithm outputs *accept*; $\psi_{i^{(h)}-1}$ does not include pk^*; for $j = [1, i-1]$, pk_j in $\psi_{i^{(h)}-1}$ is certified in $\mathcal{L}$; $|\psi_{i^{(h)}-1}| < n$. If all the conditions hold, run **Signing**$(sk^*, pk^*, m^{(h)}, \sigma', \psi_{i^{(h)}})$, and return σ and $\psi_{i^{(h)}} = \psi_{i^{(h)}-1} \parallel pk^*$. Otherwise, return $\perp$.

Output After iterating over the above steps, $\mathcal{A}$ outputs a forgery $(m^*, \sigma^*, \psi_n^*)$, where let the target signer with pk^* be the i^*th signer in ψ_n^*. Check that the following conditions hold for the given forgery: **Verification**$(m^*, \sigma^*, \psi_n^*, \{pk_i\}_{i=1}^n)$ outputs *accept*; $(m^*, i^*) \notin \{(m^{(h)}, i^{(h)})\}_{h=1}^{q_s}$ holds; ψ_n^* includes pk^*; for $j = [1, n]$, pk_j in ψ_n^* is included in $\mathcal{L}$. If all these conditions hold, then output *accept*. Otherwise, output *reject*.

Definition 5. We say that an adversary $\mathcal{A}$ breaks an ordered multisignature scheme with $(t, q_c, q_s, \ell, n, \epsilon)$ if a challenger $\mathcal{C}$ outputs *accept* in the security game described above within an execution time t and with a probability greater than ϵ. Here, $\mathcal{A}$ can generate at most q_c certification queries and at most q_s signing queries, ℓ is the length of the message output by $\mathcal{A}$, and n is the number of signers included in the forgery.

3.3 Our Basic Approach

In this section, we discuss mathematical properties to prove the security of ordered multisignatures under the standard assumptions. First, we recall the observation by Boldyreva et al. [14] as technical problems, and then describe our approach to overcome them.

3.3.1 Technical Problem

Boldyreva et al. [14] proposed the CDH-based ordered multisignature scheme based in the random oracle model, and one might think that an ordered multisignature scheme without random oracles can be constructed from their scheme. However, as described in Section 7.1 according to [14], they alleged that if the Waters multisignature scheme by Lu et al. [64] is substituted for a part of the random oracle in their ordered multisignature scheme, the approach to prove the security no longer seems to work. More specifically, a simulator for the security proof without random oracles requires a new component including random numbers to simulate the signing oracle in comparison with that with the random oracles. Here, we note that ordered multisignatures have two components, i.e., a message and the signing order. Consider that any random number included in a signature is re-utilized in the newly additional component in order to reduce the signature size. The additional component is embedded an unknown value, which is an instance of the CDH problem. Then the proof simulator has to compute with the random number, which is unknown for the simulator. Intuitively, the simulation becomes unworkable due to interference between these unknown values, i.e., the random number and the instance.

In the existing approaches to overcome this problem, a signature component is compressed into a single one by utilizing aggregate signatures. Intuitively, the approach allows the proof simulator to deal with only a single component, and hence the interference between the random number will never occur. However, the aggregate signatures bring a degradation of the efficiency since these requires a large redundant space in public parameters in order to allow each signer to deal with variable messages. Thus, we have to solve the problem without utilizing the aggregate signatures.

3.3.2 Our Strategy

Our main strategy is to utilize two properties, called the dual re-randomization and the full aggregation.

The proof in [14] is based on the re-randomization technique. Our idea is to separate an algebraic structure of the signature equation into three components, i.e., secret keys, a message and the signing order, and to give individual random number to the message component and the signing order component in order to dually execute the re-randomization. We call this technique dual re-randomization. Intuitively, the re-randomization for each component is executed in parallel. In this strategy, while the signing oracle

can be simulated as long as the re-randomization for either component is available, another random number is not affected by the random number simulating the oracle. In other words, our proof can be considered as simulations of the ordinary signatures in a dual way, and hence we can avoid the interference between the random numbers.

Meanwhile, another important property, the full aggregation, is a property such that the size of the random part in the signatures is independent of the number of signers. The full aggregation provides the efficiency and also makes a security proof easy. In particular, even if the dual re-randomization is available, the proof simulator needs component-wise reduction cost unless achieving the full aggregation. This implies that the efficiency of the security reduction is degraded in the exponential order with respect to the number of the signers. In other words, by compressing the random numbers as small as possible, a perspective of the security proof with multiple signers comes closer to a proof of ordinary signatures for a single signer, and thus the security can be efficiently proven.

Under these observations, we can construct ordered multisignature schemes without random oracles from the Waters hash function and the chameleon hash function respectively.

3.4 Waters Hash Realization

In this section, we propose an ordered multisignature scheme with the Waters hash function. We assume that there exists a trusted center to generate a public parameter.

3.4.1 The Construction

A message m in this scheme will be dealt with as a bit-string $\{0,1\}^\ell$ for all ℓ and we denote by m_i the ith bit of the message m. We can let the ℓ-bit string be the output of a collision-resistant hash functions $H : \{0,1\}^* \to \{0,1\}^\ell$.

Setup Given 1^k, generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$, random generators $g_1, g_2 \in \mathbb{G}$ and $\ell + 1$ generators $(u', u_1, \cdots, u_\ell) \in \mathbb{G}^{\ell+1}$. Output $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g_1, g_2, u', u_1, \cdots, u_\ell)$ as a public parameter *para*.

Key Generation Given $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, u', u_1, \cdots, u_\ell)$, choose random numbers $\alpha_i, t_i, v_i \leftarrow \mathbb{Z}_p^*$, and set $A_i = g_1^{\alpha_i}, T_i = g_1^{t_i}$ and $V_i = g_1^{v_i}$. Output $(g_2^{\alpha_i}, t_i, v_i)$ as a secret key sk_i and (A_i, T_i, V_i) as its corresponding public key pk_i.

Signing Given $(sk_i, pk_i, m, \sigma', \psi_{i-1})$, parse m as ℓ-bit strings $(m_1, \cdots, m_\ell) \in \{0,1\}^\ell$, σ' as $(S_{i-1}, R_{i-1}, W_{i-1})$ and ψ_{i-1} as a set $\{pk_j\}_{j\in[1,i-1]}$ of public keys, where $pk_i = (A_i, T_i, V_i)$ for all i. If $i = 1$, i.e., for the first signer in the signing group, then set $(S_{i-1}, R_{i-1}, W_{i-1}) = (1, 1, 1)$ and $\{pk_j\}_{j\in[1,i-1]} = \emptyset$ and the following verification step is skipped. Next, check that ψ_{i-1} includes pk_i. If so, output $\perp$. Otherwise, verify that the received signature σ' is a valid signature on m in ψ_{i-1} by using **Verification** for $n = i - 1$. If **Verification** outputs *reject*, abort the process and output $\perp$. Otherwise, generate random numbers $r_i, w_i \leftarrow \mathbb{Z}_p^*$ and compute as follows:

$$\begin{aligned} R_i &= R_{i-1} \cdot g_1^{r_i}, \quad W_i = W_{i-1} \cdot g_1^{w_i}, \\ S_i &= S_{i-1} \cdot g_2^{\alpha_i} \left(u' \prod_{j=1}^{\ell} u_j^{m_j} \right)^{r_i} W_i^{it_i+v_i} \left(\prod_{j\in[1,i-1]} T_j^j V_j \right)^{w_i}. \end{aligned}$$

Finally, set $\psi_i = \psi_{i-1} \parallel pk_i$, then output m, $\sigma = (S_i, R_i, W_i)$.

Verification Given $(m, \sigma, \psi_n, \{pk_i\}_{i=1}^n)$, parse m as an ℓ-bit string $(m_1, \cdots, m_\ell) \in \{0,1\}^\ell$, σ as (S_n, R_n, W_n), and extract each signer's public key (A_i, T_i, V_i) from $\{pk_i\}_{i=1}^n$. Then, check that all of $\{pk_i\}_{i=1}^n$ are distinct, and output *reject* if not. Otherwise, verify that the following equation holds:

$$\boldsymbol{e}(S_n, g_1) \stackrel{?}{=} \boldsymbol{e}\left(g_2, \prod_{i=1}^{n} A_i\right) \boldsymbol{e}\left(R_n, u' \prod_{j=1}^{\ell} u_j^{m_j}\right) \boldsymbol{e}\left(W_n, \prod_{i=1}^{n} T_i^i V_i\right).$$

If not, output *reject*. Otherwise, output *accept*.

3.4.2 Security Analysis

We now prove that the proposed scheme is secure in the standard model.

Theorem 6. The proposed ordered multisignature scheme is the $(t, q_c, q_s, \ell, n, \epsilon)$-secure if (t', ϵ')-CDH assumption in $\mathbb{G}$ holds, where

$$t' = t + (2q_c + nq_s + n)t_e, \quad \epsilon' = \frac{\epsilon}{16e(\ell+1)q_s(q_s+1)},$$

t_e is the computational cost for one exponentiation and e is base of natural logarithm.

Proof. We assume that there exists an adversary $\mathcal{A}$ who breaks the proposed scheme with $(t, q_c, q_s, \ell, n, \epsilon)$. Then, we build an algorithm $\mathcal{B}$ that solves the CDH problem. In this proof, we assume, without the loss of generality, that there exists exactly one signer, the target signer, for which $\mathcal{A}$ does not know the secret key. $\mathcal{B}$ has the list $\mathcal{L}$ of certified-keys, and for all x, we denote the value of the j-th query by $x^{(j)}$. The details are given below in **Initial Phase**. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Initial Phase: Given a challenge value (g, g^a, g^b) for the CDH problem and a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$, $\mathcal{B}$ sets $\mathcal{L} = \emptyset$, $d = 4q_s$, $g_1 = g$ and $g_2 = g^b$. Then, $\mathcal{B}$ chooses $s \leftarrow \{0, \cdots, \ell\}$, ℓ-length vectors $\boldsymbol{x_i} \leftarrow \mathbb{Z}_d^\ell$ and $\boldsymbol{y_i} \leftarrow \mathbb{Z}_p^\ell$, $x' \leftarrow \mathbb{Z}_d$, and $y' \leftarrow \mathbb{Z}_p$. Here, we define polynomials $F(m) = (p - ds) + x' + \sum_{i=1}^{\ell} x_i m_i$ and $J(m) = y' + \sum_{i=1}^{\ell} y_i m_i$. $\mathcal{B}$ also sets $u' = g_2'^{p-ds+x'} g_1^{y'}$ and $u_i = g_2'^{x_i} g_1^{y_i}$ as the generators for the public parameter, i.e., $u' \prod_{j=1}^{\ell} u_j^{m_j} = g_2^{F(m)} g^{J(m)}$. Finally, $\mathcal{B}$ generates random numbers $k^* \leftarrow [1, n]$ and $t_i \leftarrow \mathbb{Z}_p$, and then sets $T^* = (g^a)^{t^*}$, $V^* = (g^a)^{-t^* k^*} g^{v^*}$, and $A^* = g^a$ as the public key of the target signer. This means that $\mathcal{B}$ implicitly sets a value that includes ab as the target signer's signatures. Then, $\mathcal{B}$ runs $\mathcal{A}$ with $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g_1, g_2, u', u_1, \cdots, u_\ell, A_i, T_i, V_i)$.

Certification Query: For any signer, $\mathcal{A}$ generates a secret key $sk_i = (g_2^{\alpha_i}, t_i, v_i)$ and its corresponding public key $pk = (A_i, T_i, V_i)$, and then provides sk_i and pk_i to $\mathcal{B}$. $\mathcal{B}$ checks that $\boldsymbol{e}(g_2^{\alpha_i}, g_1) = \boldsymbol{e}(g_2, A_i)$, $V_i = g_1^{v_i}$, and $T_i = g_1^{t_i}$. If all these equations hold, pk_i is registered in $\mathcal{L}$. Otherwise, the output is $\perp$.

Signing Query: Given a signing query $(pk^*, m^{(h)}, \psi_{i^{(h)}-1}, \sigma')$ generated by $\mathcal{A}$ for the target signer, $\mathcal{B}$ checks that $F(m^{(h)}) \neq 0 \vee i^{(h)} \neq k^*$ holds where $i^{(h)}$ is a position of the target signer for h-th query, and aborts the process if the condition does not hold. Otherwise, $\mathcal{B}$ chooses random numbers $(r, w) \leftarrow \mathbb{Z}_p$ and computes either one of the following cases:

(Case 1) $F(m^{(h)}) \neq 0$: Compute as follows:

$$
\begin{aligned}
R_{i^{(h)}} &= g^r (g^a)^{-\frac{1}{F(m^{(h)})}}, \quad W_{i^{(h)}} = g^w, \\
S_{i^{(h)}} &= (g^a)^{-\frac{J(m^{(h)})}{F(m^{(h)})}} \left(u' \prod_{j=1}^{\ell} u_j^{m_j} \right)^r \left(\prod_{j=1}^{i^{(h)}} (T_j^j V_j) \right)^w g_2^{\sum_{j \in [1, i^{(h)}-1]} \alpha_j}.
\end{aligned}
$$

From **Initial Phase**, the following equation holds:

$$\begin{aligned} S_{i^{(h)}} &= g^{ab}\left((g^b)^{F(m^{(h)})}g^{J(m^{(h)})}\right)^{-\frac{a}{F(m^{(h)})}}\left(u'\prod_{j=1}^{\ell}u_j^{m_j}\right)^r\left(\prod_{j=1}^{i^{(h)}}T_j^jV_j\right)^w \\ &\quad \times g_2^{\sum_{j\in[1,i^{(h)}-1]}\alpha_j} \\ &= g_2^{a+\sum_{j\in[1,i^{(h)}-1]}\alpha_j}\left(u'\prod_{j=1}^{\ell}u_j^{m_j}\right)^{r-\frac{a}{F(m^{(h)})}}\left(\prod_{j=1}^{i^{(h)}}T_j^jV_j\right)^w. \end{aligned}$$

This (S_i, R_i, W_i) becomes a valid signature.

(Case 2) $F(m^{(h)}) = 0 \wedge i^{(h)} \neq k^*$: Compute as follows:

$$\begin{aligned} R_{i^{(h)}} &= g^r, \quad W_{i^{(h)}} = g^w(g^b)^{-\frac{1}{t^*(i^{(h)}-k^*)}}, \\ S_{i^{(h)}} &= (g^b)^{-\frac{v^*}{t^*(i^{(h)}-k^*)}}\left((g^a)^{i^{(h)}t^*}(g^a)^{-t^*k^*}g^{v^*}\right)^w\left(u'\prod_{j=1}^{\ell}u_j^{m_j}\right)^r \\ &\quad \times (W_{i^{(h)}})^{\sum_{j\in[1,i^{(h)}-1]}(jt_j+v_j)}g_2^{\sum_{j\in[1,i^{(h)}-1]}\alpha_j}. \end{aligned}$$

From **Initial Phase**, the following equation holds:

$$\begin{aligned} S_{i^{(h)}} &= g^{ab}g^{-ab\frac{t^*(i^{(h)}-k^*)}{t^*(i^{(h)}-k^*)}}(g^b)^{-\frac{v^*}{t^*(i^{(h)}-k^*)}}\left((g^a)^{i^{(h)}t^*}(g^a)^{-t^*k^*}g^{v^*}\right)^w\left(u'\prod_{j=1}^{\ell}u_j^{m_j}\right)^r \\ &\quad \times (W_{i^{(h)}})^{\sum_{j\in[1,i^{(h)}-1]}jt_j+v_j}g_2^{\sum_{j\in[1,i^{(h)}-1]}\alpha_j} \\ &= g^{ab}\left(g^{at^*(i^{(h)}-k^*)+v^*}\right)^{-\frac{b}{t^*(i^{(h)}-k^*)}}\left((g^a)^{i^{(h)}t^*}(g^a)^{-t^*k^*}g^{v^*}\right)^w\left(u'\prod_{j=1}^{\ell}u_j^{m_j}\right)^r \\ &\quad \times (W_{i^{(h)}})^{\sum_{j\in[1,i^{(h)}-1]}jt_j+v_j}g_2^{\sum_{j\in[1,i^{(h)}-1]}\alpha_j} \\ &= g^{ab}\left((T^*)^{i^{(h)}}V^*\right)^{w-\frac{b}{t^*(i^{(h)}-k^*)}}\left(u'\prod_{j=1}^{\ell}u_j^{m_j}\right)^r \\ &\quad \times (W_{i^{(h)}})^{\sum_{j\in[1,i^{(h)}-1]}jt_j+v_j}g_2^{\sum_{j\in[1,i^{(h)}-1]}\alpha_j} \\ &= g_2^{a+\sum_{j\in[1,i^{(h)}-1]}\alpha_j}\left(u'\prod_{j=1}^{\ell}u_j^{m_j}\right)^r\left(\prod_{j=1}^{i^{(h)}}T_j^jV_j\right)^{w-\frac{b}{t^*(i^{(h)}-k^*)}}. \end{aligned}$$

This (S_i, R_i, W_i) becomes also a valid signature.

Output: After iterating over the steps described above, $\mathcal{A}$ outputs a forgery $\sigma^* = (S_n^*, R_n^*, W_n^*)$ on a message m^* in the signing order ψ_n^*.

If $F(m^*) \neq 0 \vee i^* \neq k^*$ holds in the forgery output by $\mathcal{A}$ where i^* is the position of the target signer, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ can solve the CDH problem via the forgery by $\mathcal{A}$ as follows. If the verification equation holds and $F(m^*) = 0 \wedge i^* = k^*$ holds, then S^* can be written as follows:

$$\begin{aligned} S^* &= g_2^{a+\sum_{i=1\wedge i\neq i^*}^n \alpha_i} \left((g^b)^{F(m^*)} g^{J(m^*)}\right)^r \left((g^a)^{t^*(i^*-k^*)} g^{v^*} \prod_{i=1\wedge i\neq i^*}^n T_i^i V_i\right)^w \\ &= g_2^{a+\sum_{i=1\wedge i\neq i^*}^n \alpha_i} \left(g^{J(m^*)}\right)^r \left(g^{v^*} \prod_{i=1\wedge i\neq i^*}^n T_i^i V_i\right)^w . \end{aligned}$$

Then, g^{ab} can be obtained from

$$\frac{S^*}{g_2^{\sum_{j=1\wedge j\neq i^*}^n \alpha_j} (R^*)^{J(m^*)} (W^*)^{v^*+\sum_{i=1\wedge i\neq i^*}^n (it_i+v_i)}}.$$

$\mathcal{B}$ knows all the secret values except for ab and so can compute the above values. If $\mathcal{B}$'s guess is correct, $\mathcal{B}$ can solve the CDH problem. Therefore, the probability is given as follows:

$$\begin{aligned} \epsilon' &\geq \epsilon \cdot \Pr[E_1] \Pr[E_2], \\ E_1 &= \left[\left(\bigwedge_{h=1}^{q_s} F(m^{(h)}) \neq 0\right) \wedge F(m^*) = 0\right], \\ E_2 &= \left[\bigwedge_{h=1}^{q_s} \left(i^{(h)} \neq k^*\right) \wedge i^* = k^*\right], \end{aligned}$$

and $i^{(h)}$ is the position of the target signer for the signing queries. E_1 is analyzed in a manner similar to the proof in [95], and E_2 is analyzed in a manner similar to the proof in [14]. Therefore, $\Pr[E_1] = \frac{1}{16q_s(\ell+1)}$ and $\Pr[E_2] = \frac{1}{e(q_s+1)}$. Thus, the probability is $\epsilon' = \frac{\epsilon}{16e(\ell+1)q_s(q_s+1)}$. Additionally, the execution time of $\mathcal{B}$ is that of $\mathcal{A}$ plus two exponentiation computations for **Certification Query**, n exponentiation computations for **Signing Query** for q_s times, and n exponentiations for the final step. Therefore, $t' = t + (2q_c + nq_s + n)t_e$ holds, where t_e is the computational time for one exponentiation computation. □

3.5 Chameleon Hash Realization

In this section, we propose an ordered multisignature scheme with the chameleon hash function. This scheme utilizes the synchronized setting [1, 36], and the data size of the public parameter becomes smaller than the scheme in the previous section.

3.5.1 The Construction

A message m in this scheme will be dealt with as an element m in $\mathbb{Z}_p$, and we can also utilize a collision-resistant hash functions $H : \{0,1\}^* \to \{0,1\}^k$ to obtain m. In this scheme, we assume that signers are given the value $s = \mathbf{clock}()$ as a common input to the following signing algorithm. It keeps as internal state s_{prev} denoting the last time period on which it issued a signature.

Setup Given 1^k, generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$, seven generators $(g, u, h, d, c, z, y) \in \mathbb{G}^7$, and a pseudo-random function (PRF) key K. Output $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, u, h, d, c, z, y, K)$ as a public parameter *para* where $PRF : \{0,1\}^* \to \mathbb{Z}_p$ be a pseudo-random function family.

Key Generation Given $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, u, h, d, c, z, y)$, choose random numbers $\alpha_i, t_i, v_i \leftarrow \mathbb{Z}_p^*$, and set $A_i = g^{\alpha_i}, T_i = g^{t_i}$ and $V_i = g^{v_i}$. Output (α_i, t_i, v_i) as a secret key sk_i and (A_i, T_i, V_i) as its corresponding public key pk_i.

Signing Given $(sk_i, pk_i, m, \sigma', \psi_{i-1})$, parse σ' as $(S_{i-1}, R_{i-1}, W_{i-1}, \{x_j\}_{j=1}^{i-1}, s)$ and ψ_{i-1} as a set $\{pk_j\}_{j=1}^{i-1}$ of public keys, where $pk_i = (A_i, T_i, V_i)$ for all i. If $i = 1$, i.e., for the first signer in the signing group, then set $(S_{i-1}, R_{i-1}, W_{i-1}) = (1, 1, 1)$, $[1, j-1] = \emptyset$, $s = \mathbf{clock}()$ and $x := PRF_K(m)$, and the following verification step is skipped. Next, check that ψ_{i-1} includes pk_i and that $s_{prev} = s$ or $s \geq 2^k$. If so, output $\perp$. Otherwise, verify that the received signature σ' is a valid signature on m in ψ_{i-1} by using **Verification** for $n = i - 1$. If **Verification** outputs *reject*, abort the process and output $\perp$. Otherwise, record the current time period as $s_{prev} := s$, generate two random numbers $(r_i, w_i) \leftarrow \mathbb{Z}_p^*$. Then, compute as follows:

$$
\begin{aligned}
R_i &= R_{i-1} \cdot g^{r_i}, \quad W_i = W_{i-1} \cdot g^{w_i}, \\
S_i &= S_{i-1} \cdot (u^m h^x d)^{\alpha_i} \left(c^{\lceil \lg(s) \rceil} z^s y \right)^{r_i} W_i^{it_i + v_i} \left(\prod_{j \in [1, i-1]} T_j^j V_j \right)^{w_i}.
\end{aligned}
$$

Finally, set $\psi_i = \psi_{i-1} \parallel pk_i$, then output m, $\sigma = (S_i, R_i, W_i, x, s)$.

Verification Given $(m, \sigma, \psi_n, \{pk_i\}_{i=1}^n)$, parse σ as (S_n, R_n, W_n, x, s), and extract each signer's public key (A_i, T_i, V_i) from $\{pk_i\}_{i=1}^n$. Then, check that all of $\{pk_i\}_{i=1}^n$ are distinct, and output *reject* if not. Then, check that $2^k > s > 0$ and $m = PRF_K(m)$, and output *reject* if not. Otherwise, verify that the following equation holds:

$$\boldsymbol{e}(S_n, g) \stackrel{?}{=} \boldsymbol{e}\left(u^m h^x d, \prod_{i=1}^n A_i\right) \boldsymbol{e}\left(c^{\lceil \lg(s) \rceil} z^s d, R_n\right) \boldsymbol{e}\left(\prod_{i=1}^n T_i^i V_i, W_n\right).$$

If not, output *reject*. Otherwise, output *accept*.

3.5.2 Security Analysis

We now prove that the proposed scheme is secure in the standard model. The proof is similar to that of the Theorem 6 described in Section 3.4.

Theorem 7. The proposed ordered multisignature scheme is the $(t, q_c, q_s, \ell, n, \epsilon)$-secure if (t', ϵ')-CDH assumption in $\mathbb{G}$ holds, where

$$t' = t + (2q_c + (6 + 5n)q_s + n + 2)t_e, \quad \epsilon' = \frac{\epsilon}{e^2(q_s + 1)^2},$$

t_e is the computational cost for one exponentiation and e is base of natural logarithm.

Proof. In this proof, the setting is almost the same as that in Theorem 4 except that The main strategy of the proof is based on the Hohenberger-Waters technique [42]. In particular, $\mathcal{B}$ first guess a value l^* in the range 1 to k where k is the security parameter. This represents a guess that $\mathcal{A}$ will forge on some index s such that $l^* = \lceil \lg(s) \rceil$. In the original proof in [42], the proof consists of two steps, i.e., $l^* = s$ and $l^* = \lceil \lg(s) \rceil$. However, in this proof, we can consider only the case of $l^* = \lceil \lg(s) \rceil$. The reason is that signatures can be simulated by either component in comparison with the proof in [42]. Even if l^* which is one of the abort conditions of the proof in [42] holds, the proof in this section works via the re-randomization of the signing order part, i.e., $i^{(h)} \neq k^*$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Initial Phase: Given a challenge value (g, g^a, g^b) for the CDH problem and a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$, $\mathcal{B}$ sets $\mathcal{L} = \emptyset$, and $d = g^b$, and chooses $l^* \leftarrow [1, k]$, $k^* \leftarrow [1, n]$ and $(x_u, x_h, x_z, x_c, x_y) \leftarrow \mathbb{Z}_p$. $\mathcal{B}$ also picks a PRF key K and sets $u = g^{x_u}, h = g^{x_h}, z = g^{x_z}$, $c = g^b g^{x_c}$ and

$y = (g^b)^{-l^*} g^{x_y}$ as the public parameter. Then, $\mathcal{B}$ sets the public key of the target signer similarly as the proof of Theorem 4. Finally, $\mathcal{B}$ runs $\mathcal{A}$ with $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, u, h, d, c, z, y, K, A_i, T_i, V_i)$.

Certification Query: This step is exactly the same as that in Theorem 4.

Signing Query: This step is almost the same as the proof in Theorem 4, but in this proof the condition that $\mathcal{B}$ aborts is whether $\lceil \lg(s^{(h)}) \rceil \neq l^* \vee i^{(h)} \neq k^*$ holds or not. Given a query $(pk^*, m^{(h)}, \psi_{i^{(h)}-1}, \sigma')$, $\mathcal{B}$ aborts the process if the condition does not hold. Otherwise, $\mathcal{B}$ chooses random numbers $(r, w) \leftarrow \mathbb{Z}_p$ and computes either one of the following cases:

(Case 1) $\lceil \lg(s^{(h)}) \rceil \neq l^*$: Compute as follows:

$$
\begin{aligned}
R_{i^{(h)}} &= g^r (g^a)^{\frac{1}{\lceil \lg(s^{(h)}) \rceil - l^*}}, \quad W_{i^{(h)}} = g^w, \\
S_{i^{(h)}} &= (g^a)^{x_u m + x_h x} \left(g^r (g^a)^{\frac{1}{\lceil \lg(s^{(h)}) \rceil - l^*}} \right)^{x_c \lceil \lg(s^{(h)}) \rceil + x_z s^{(h)} + x_y} \\
&\quad \times (g^b)^{r\left(\lceil \lg(s^{(h)}) \rceil - l^*\right)} \cdot \left(\prod_{j=1}^{i^{(h)}} (T_j^j V_j) \right)^w \prod_{j=1}^{i^{(h)}-1} (u^m h^x d)^{\alpha_j} .
\end{aligned}
$$

From **Initial Phase**, the following equation holds where $r' := r - \frac{a}{\lceil \lg(s^{(h)}) \rceil - l^*}$:

$$
\begin{aligned}
S_{i^{(h)}} &= (u^m h^x d)^a \left(g^{x_c \lceil \lg(s^{(h)}) \rceil} z^s \right)^{r'} (g^{x_y})^{r'} g^{ab} g^{-ab \frac{\lceil \lg(s^{(h)}) \rceil - l^*}{\lceil \lg(s^{(h)}) \rceil - l^*}} \\
&\quad \times (g^b)^{r\left(\lceil \lg(s^{(h)}) \rceil - l^*\right)} \left(\prod_{j=1}^{i^{(h)}} (T_j^j V_j) \right)^w \prod_{j=1}^{i^{(h)}-1} (u^m h^x d)^{\alpha_j} \\
&= (u^m h^x d)^a \left(g^{x_c \lceil \lg(s^{(h)}) \rceil} z^s \right)^{r'} (g^{x_y})^{r'} g^{b\left(\lceil \lg(s^{(h)}) \rceil - l^*\right)\left(r - \frac{a}{\lceil \lg(s^{(h)}) \rceil - l^*}\right)} \\
&\quad \times \left(\prod_{j=1}^{i^{(h)}} (T_j^j V_j) \right)^w \prod_{j=1}^{i^{(h)}-1} (u^m h^x d)^{\alpha_j} \\
&= \prod_{j=1}^{i^{(h)}-1} (u^m h^x d)^{\alpha_j} (u^m h^x d)^a \left(c^{\lceil \lg(s^{(h)}) \rceil} z^{s^{(h)}} y \right)^{r'} \left(\prod_{j=1}^{i^{(h)}} (T_j^j V_j) \right)^w .
\end{aligned}
$$

These values become a valid signature.

(Case 2) $\lceil \lg(s^{(h)}) \rceil = l^* \wedge i^{(h)} \neq k^*$: Compute as follows:

$$\begin{aligned}
R_{i^{(h)}} &= g^r, \quad W_{i^{(h)}} = g^w (g^b)^{-\frac{1}{t^*(i^{(h)}-k^*)}}, \\
S_{i^{(h)}} &= (g^a)^{x_u m} (g^a)^{x_u x} (g^b)^{-\frac{v^*}{t^*(i^{(h)}-k^*)}} \left((g^a)^{i^{(h)} t^*} (g^a)^{-t^* k^*} g^{v^*} \right)^w \\
&\quad \times \left(c^{\lceil \lg(s^{(h)}) \rceil} z^{s^{(h)}} y \right)^r (W_{i^{(h)}})^{\sum_{j \in [1, i^{(h)}-1]} (j t_j + v_j)} \left(\prod_{j=1}^{i^{(h)}} (T_j^j V_j) \right)^w \\
&\quad \times \prod_{j=1}^{i^{(h)}-1} (u^m h^x d)^{\alpha_j} .
\end{aligned}$$

The computation is almost the same as that of Theorem 4, and we skip the detail of the computation. These values become also a valid signature.

Output: After iterating over the steps described above, $\mathcal{A}$ outputs a forgery $\sigma^* = (S_n^*, R_n^*, W_n^*, x^*, s^*)$ on a message m^* with signing order ψ_n^*.

If $l^* \neq \lceil \lg(s^*) \rceil \vee i^* \neq k^*$ holds in the forgery output by $\mathcal{A}$ where i^* is the position of the target signer, then $\mathcal{B}$ aborts. Otherwise, similarly as the proof of Theorem 4, $\mathcal{B}$ can solve the CDH problem via the forgery by $\mathcal{A}$. In particular, if the verification equation and $l^* = \lceil \lg(s^*) \rceil \wedge i^* = k^*$ hold, then S^* can be written as follows where α_{i^*} corresponds to a:

$$\begin{aligned}
S^* &= \left((g^{x_u})^{m^*} (g^{x_h})^{x^*} g^b \right)^a \prod_{i=1 \wedge i \neq i^*}^{n} \left(u^{m^*} h^{x^*} d \right)^{\alpha_j} \\
&\quad \times \left(\left(g^b g^{x_c} \right)^{\lceil \lg(s^*) \rceil} g^{x_z s^*} \left(g^{-b l^*} g^{x_y} \right) \right)^r \left((g^a)^{t^*(i^*-k^*)} g^{v^*} \prod_{i=1 \wedge i \neq i^*}^{n} T_i^i V_i \right)^w \\
&= \left((g^{x_u})^{m^*} (g^{x_h})^{x^*} g^b \right)^a \prod_{i=1 \wedge i \neq i^*}^{n} \left(u^{m^*} h^{x^*} d \right)^{\alpha_j} \left((g^{x_c})^{\lceil \lg(s^*) \rceil} g^{x_z s^*} (g^{x_y}) \right)^r \\
&\quad \times \left(g^{v^*} \prod_{i=1 \wedge i \neq i^*}^{n} T_i^i V_i \right)^w .
\end{aligned}$$

Then, g^{ab} can be obtained from

$$\frac{S^* / \prod_{i=1 \wedge i \neq i^*}^{n} \left(u^{m^*} h^{x_i^*} d \right)^{\alpha_j}}{(g^a)^{x_u m^* + x_h x_{i^*}^*} (R^*)^{x_c \lceil \lg(s^*) \rceil + x_z s^* + x_y} (W^*)^{v^* + \sum_{i=1 \wedge i \neq i^*}^{n} (i t_i + v_i)}} .$$

The success probability and the execution time of $\mathcal{B}$ can be obtained similarly as the proof of Theorem 4, where the probability is as follows:

$$\Pr\left[\bigwedge_{h=1}^{q_s}(\lceil \lg(s^*)\rceil \neq k) \wedge (i^* = k^*) \wedge \bigwedge_{h=1}^{q_s}\left(i^{(h)} \neq k^*\right) \wedge (i^* = k^*)\right].$$

The probability can be analyzed as E_2 in the proof of Theorem 4, and therefore the probability is $\epsilon' = \frac{\epsilon}{e^2(q_s+1)^2}$. Similarly, the execution time of $\mathcal{B}$ is $t' = t + (2q_c + (6+5n)q_s + n + 2)\, t_e$ holds, where t_e is the computational time for the exponentiation computation. □

3.6 Discussion

In this section, we compare the performance of the proposed schemes with the LOSSW06 scheme, the AGH10 scheme, and ECDSA as a naive approach in terms of the memory size of routers, the communication cost and the verification time.

We evaluate the memory size by summations of the public parameter, a signature and public keys of the signers, where the number of the signers is at most 60000. As described in Section 7.1, there are 36000 ASes, and the maximized value for 16-bit AS number is 65535. Hence, we should evaluate the memory size for such a scale. Let the security parameter be 128-bit security, i.e., the size of an element in $\mathbb{G}$ is 256 bits. Although the authors of the AGH10 scheme described the Naccache approach [70] to reduce the parameter size, the approach brings down the efficiency of the security reduction. Hence, we do not consider the Naccache approach. Under these conditions, we give the result in Fig.3.1(a).

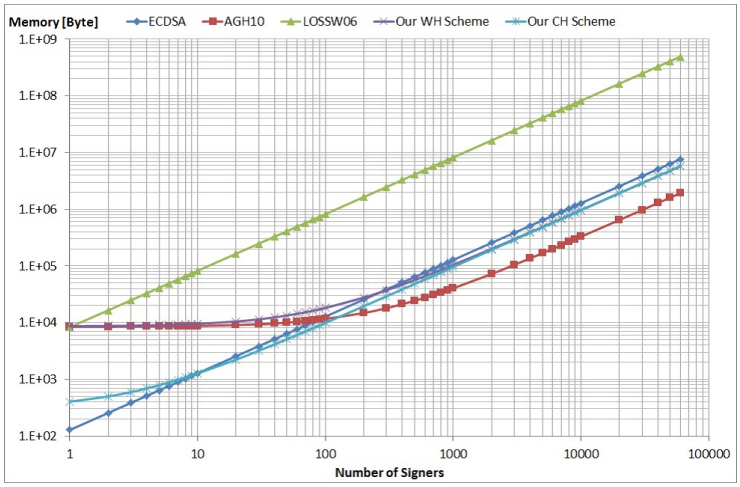
For the communication cost, we evaluate the performances by summations of a single ordered multisignature and subject key identifier of the public keys, where the size of the subject key identifier is based on the security level. Here, let the number of signers be 20. The reason of 20 signers is, as shown in [49], because a distance between any two ASes can be fully covered by 20 hops. We give Fig.3.1(b) as the detail.

For the verification time, we utilize the benchmark of PBC library [65] as a typical value, where the values we referred to here is the type A curve shown in the TEPLA web page [47]. According to [47], the computational time per a bilinear map is 2.2 msec and the computational time per a scalar multiplication, i.e., an exponentiation, is 2.55 msec. These values are on 80-bit security which is different from the above setting, because the existing library providing symmetric pairings is only the type A curve in PBC library.

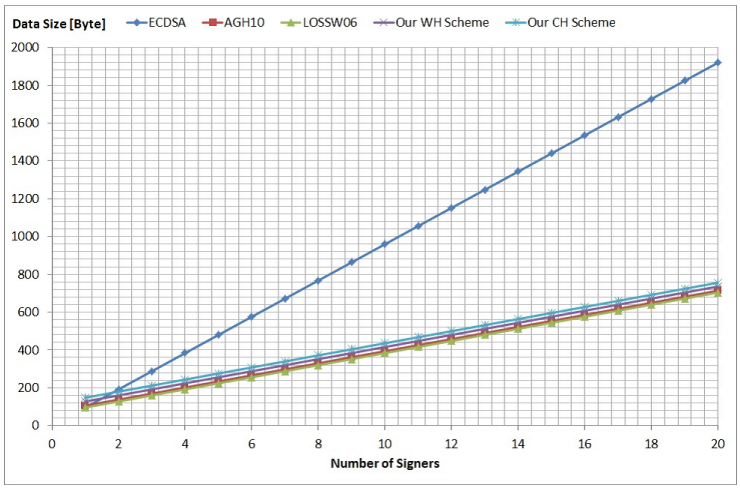
The evaluation of the verification is for an expected value rather than a measured value. We give Fig.3.1(c) as the estimation result.

Under these conditions, we conclude our schemes are better than the other schemes consisting of aggregate signatures. The LOSSW06 scheme requires a large amount of memory, and this scheme is unavailable in routers. The required size of memory for our schemes are less than one-percent relatively to that of the LOSSW06 scheme. Meanwhile, in the AGH10 scheme, the total computational delay is quite large. In general, we can consider only three hops, i.e., Internet service provider (ISP), national Internet registry (NIR) and ISP. Even in such a network, the delay by the AGH10 scheme is larger than one-sec, and so this scheme is impractical.

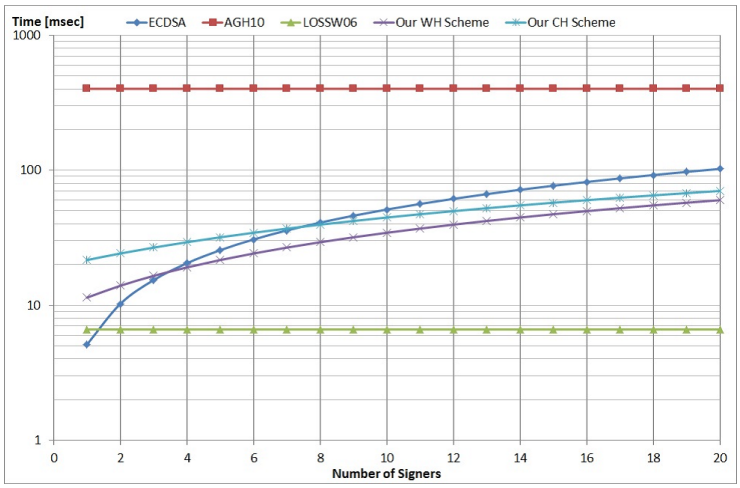
In our schemes, the chameleon-hash-based scheme requires smaller memory than the Waters-hash-based scheme for a small number of signers. However, the chameleon-hash-based scheme also requires state-information, and the security cannot be proven if the same state-information is re-used. Hence, the Waters-hash-based scheme is based on more standard assumptions.



(a) Size of Required Memory of the Schemes



(b) Communication Cost of the Schemes



(c) Verification Time of the Schemes

Figure 3.1: Loads of Routers: We evaluate the performances of our schemes with the existing schemes, ECDSA, the AGH10 scheme and the LOSSW06 scheme. We denote by Our WH Scheme as the scheme based on the Waters hash function, and by Our CH scheme as the scheme based on the chameleon hash function.

Chapter 4

Structured Multisignatures

4.1 Introduction

4.1.1 Motivation

Verifying the signing order is sometimes desirable in multisignatures, and ordered multisignatures described in Chapter 3 is such a primitive. However, in the real world, relationship among a group of signers is more complicated. For example, for a group in a company consisting of a boss and his/her subordinates, the boss should sign a document after the subordinates have signed it. A scheme providing the verification of such a complicated structure is called *structured multisignature* scheme [30]. This primitive is an extension of the ordered multisignatures and has been expected. Many such schemes have been proposed so far [19, 30, 48, 61, 62, 72, 90, 94, 102]. Whereas the history of structured multisignature scheme is long, there are few schemes with the provable security. To the best of our knowledge, only the scheme by Tada [90], called Tada03 scheme for short, achieves the provable security with the reduction to DLP. However, the Tada03 scheme has a restriction in the number of signers to guarantee the security. In particular, there is an attack called attack-0 [90] that signatures of any signer can be forged via an algebraic structure representing the signing group. In this chapter, we construct a structured multisignature scheme without such a restriction.

4.1.2 Our Construction

In this chapter, we propose a structured multisignature scheme, which is provably secure under DLP assumption, without restriction in the number of signers. Our basic idea to remove the restriction is to utilize a ring homo-

morphism. The vulnerability of the Tada03 scheme against the attack-0 is due to its algebraic structure: more precisely, the signature equation in the Tada03 scheme includes some coefficient depending on a signing group and an adversary can manipulate the coefficient to forge a signature. In order to resist this attack, we extend the algebraic structure to the ring homomorphism. Ring is an extended notion of group and a cryptographic scheme based on the ring homomorphism provides more possible applications. We consider that the ring homomorphism allows us to remove the coefficient developed in the Tada03 scheme, and thus can construct a structured multisignature scheme without the restriction. In the proof of the security, we adopt to prove in the random oracle model. Through this proof, we adopt the same strategy with the paper [90] to prove the security. In this strategy, we have reduce the security of the structured multisignature scheme into the multi-round identification scheme.

As an instantiation of the proposed scheme, we adopt the non-commutative ring homomorphism by Chida et al. [26]. In recent years, homomorphic encryption based on algebraic structure such as ring homomorphism is anticipated for securing cloud computing and is extensively studied [92, 36, 87]. It is an interesting question whether ring homomorphism can be positively applied to other applications such as multisignature schemes. However, these are inefficient, and the ring homomorphism in [26] consists of simple operations utilizing matrix. Thus, we focus on only the ring homomorphism in [26].

Through this chapter, we define two signer structures, parallel structure [19] and serial structure [19]. Intuitively, parallel structure means that the signing order does not have any meaning, and serial structure means that the signing order have some meaning. So, in the example of the group between the boss and the subordinates, we say the structure between the boss and the subordinate is serial structure. Similarly, we say the structure among the subordinates is parallel structure. Our proposed scheme uses non-commutative operation for the serial structure and commutative operation for the parallel structure.

4.1.3 Related Work

Keys in the first multisignature scheme by Itakura et al. [48] are only valid for restricted structures of signers. Later Okamoto [74] proposed the first multisignature scheme which specifies the signing order of a group of signers without any restriction on the structure of signers. However, these schemes do not truly the signing group. The first structured multisignature scheme

was proposed by Doi et al. [30]. After, many structured multisignature schemes have been proposed in [19, 61, 62, 72, 90, 102, 94]. They are roughly classified into two categories, the DLP-based, including the bilinear map, schemes [19, 61, 62, 72, 90, 102, 94] and the permutation-based scheme [30]. Among them, the permutation-based scheme is unsuitable in complicated structures such as hierarchical structure and requires a large parameter in comparison with the DLP-based construction. Hence, we discuss the DLP-based construction in this work.

In DLP-based scheme, the scheme by Burmester et al. [19], called the BDD+00 scheme for short, the scheme by Li et al. [61], called the LZL04 scheme for short, and the scheme by Lin et al. [62], the LWZ03 scheme for short, have no order flexibility described in the paper [68], and hence these are more restricted in the number of signers in comparison with that in the Tada03 scheme. The security of the scheme by Wang et al. [94], called the WOMOD07 scheme for short, and that of the scheme by Ohmori et al. [72], called the OCSN96 scheme for short, are not rigorously proven in the sense of no reduction algorithm. Hence, the best exsiting scheme scheme is the Tada03 scheme. However, the security of the Tada03 scheme is proven in a case that the number of the signers is restricted in order to prevent the attack-0 described in [90].

4.2 Non-commutative Ring Homomorphism

We define a group homomorphism and a ring homomorphism, and then explain the non-commutative ring homomorphism by Chida et al. [26].

4.2.1 Homomorphism

Definition of Group Homomorphism Let $(\mathbb{G}_1, \circ)$ and $(\mathbb{G}_2, \bullet)$ be groups. A (group) homomorphism is a function $f : \mathbb{G}_1 \to \mathbb{G}_2$ from $\mathbb{G}_1$ to $\mathbb{G}_2$ for which the following condition hold:

$$\forall a, b \in \mathbb{G}_1,\ f(a \circ b) = f(a) \bullet f(b).$$

Definition of Ring Homomorphism Let $(\mathbb{R}_1, +, *)$ and $(\mathbb{R}_2, \dagger, \circ)$ be rings. A (ring) homomorphism is a function $f : \mathbb{R}_1 \to \mathbb{R}_2$ from $\mathbb{R}_1$ to $\mathbb{R}_2$ for which the following condition hold:

$$\forall a, b \in \mathbb{R}_1,\ f(a + b) = f(a) \dagger f(b),\ f(a * b) = f(a) \circ f(b)$$

4.2.2 Non-commutative Ring Homomorphism by Chida et al.

We recall the ring homomorphism by Chida et al. [26]. First, we recall operations and then recall a function.

Definition of Operations Let $\mathbb{R}$ and $\mathbb{S}$ be finite rings which consist of the following conditions:

$$\mathbb{R} = \begin{pmatrix} \mathbb{Z}_{p-1} & \mathbb{Z}_{p-1} \\ 0 & \mathbb{Z}_{p-1} \end{pmatrix} \text{ and } \mathbb{S} = \begin{pmatrix} \mathbb{Z}_{p-1} & \langle g \rangle \\ 0 & \mathbb{Z}_{p-1} \end{pmatrix}, \tag{4.1}$$

where p is a large prime and $g \in \mathbb{Z}_p^*$. Addition '$\dot{+}$', multiplication '$\odot$' and addition '$\dagger$', multiplication '$\circ$' are defined in rings $\mathbb{R}, \mathbb{S}$, respectively. We denote them by $(\mathbb{R}, \dot{+}, \odot), (\mathbb{S}, \dagger, \circ)$. Those operations are defined as follows:

$$\dot{+} : \begin{pmatrix} a_1 & x \\ 0 & a_2 \end{pmatrix} \dot{+} \begin{pmatrix} b_1 & y \\ 0 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 + b_1 & x + y \\ 0 & a_2 + b_2 \end{pmatrix} \tag{4.2}$$

$$\odot : \begin{pmatrix} a_1 & x \\ 0 & a_2 \end{pmatrix} \odot \begin{pmatrix} b_1 & y \\ 0 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 \cdot b_1 & a_1 \cdot y + b_2 \cdot x \\ 0 & a_2 \cdot b_2 \end{pmatrix} \tag{4.3}$$

$$\dagger : \begin{pmatrix} a_1 & x \\ 0 & a_2 \end{pmatrix} \dagger \begin{pmatrix} b_1 & y \\ 0 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 + b_1 & x \cdot y \\ 0 & a_2 + b_2 \end{pmatrix} \tag{4.4}$$

$$\circ : \begin{pmatrix} a_1 & x \\ 0 & a_2 \end{pmatrix} \circ \begin{pmatrix} b_1 & y \\ 0 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 \cdot b_1 & y^{a_1} \cdot x^{b_2} \\ 0 & a_2 \cdot b_2 \end{pmatrix} \tag{4.5}$$

Definition of Function A concrete one-way function $F : \mathbb{R} \to \mathbb{S}$ is defined as follows:

$$F\left(\begin{pmatrix} a & b \\ 0 & c \end{pmatrix}\right) \equiv \begin{pmatrix} a \bmod p-1 & g^b \bmod p \\ 0 & c \bmod p-1 \end{pmatrix} \tag{4.6}$$

The one-way function $F : \mathbb{R} \to \mathbb{S}$ is shown to be a non-commutative ring homomorphism in [26].

4.3 Structured Multisignature Scheme

In this section, we define structures of signers, a syntax of structured multisignature scheme and the security model.

4.3.1 Signer Structure

In this subsection, we define structures of signers. We define two structures as the relation for a group of signers, serial structure and parallel structure, and consider a series-parallel graph described in Chapter 2 as a signer structure. Here, an edge of a series-parallel graph corresponds to a signer, and a unique edge for the graph corresponds to a unique index representing the position of each signer in any structure, i.e., an index i in ψ_n corresponds to pk_i which is the ith signer in the signer structure. We denote by ψ_n a signer structure consisting of n signers, by $\mathcal{T}(i)$ a set of signers connecting to the initial vertex I_i of ith signer in a way such that $\mathcal{T}(i) = \{x | I_i = T_x \wedge 1 \leq x < i \wedge G_x(I_x, T_x) \subset \psi_n\}$, by $\mathcal{I}(i)$ a set of signers connecting to the terminal vertex T_i of ith signer in a way such that $\mathcal{I}(i) = \{x | T_i = I_x \wedge i < x \leq n \ \wedge G_x(I_x, T_x) \subset \psi_n\}$, by $\{a_j\}_{j \in \mathcal{T}(i)}$, for all a, all a_j for $j \in \mathcal{T}(i)$. We also define an operation $\dot{\subset}$ where $i \dot{\subset} \psi_n$ means extractions of indexes from ψ_n.

4.3.2 Syntax

In this subsection, we introduce a syntax of structured multisignature scheme. We assume that each signer pk_i has each hash function f_i and h_i as random oracles.

Setup Given a security parameter 1^k as input, output a public parameter $param$.

Key Generation Given $para$, output a secret key sk_i and its corresponding public key pk_i.

Signing Given a message m, a structure ψ_i, signatures σ_j for $j \in \mathcal{T}(i)$ and sk_i as input, generate a multisignature σ_i on m for a structure ψ_i. Output σ_i.

Verification Given $(m, \psi_i, \{pk_j\}_{j=1}^{i}, \sigma_i)$ as input, output $accept$ or $reject$.

4.3.3 Security of Structured Multisignature Scheme

In this subsection, we introduce the notion of the security of the structured multisignature scheme. This model have been proposed in [90]. We adopt this model as adversary model to prove the security. In this model, we allow an adversary $\mathcal{A}$ to collude signers to obtain signatures from them.

Adversary Given a public parameter *param* and public keys $\{pk_i\}_{i=1}^n$ of all signers, an adversary $\mathcal{A}$ who has access to random oracles F_i and H_i for $i \in [1, n]$ executes the following steps at most h times.

1. $\mathcal{A}$ chooses i signers and then generate a structure $\psi_i^{(h)}$ generated by i signers and a message $m^{(h)}$.
2. $\mathcal{A}$ generates a signature $\sigma_{j-1}^{(h)}$ in the structure $\psi_j^{(h)}$ by colluding signers, where $j \in \mathcal{T}(i)$, and then asks the i-th signer to sign $m^{(h)}$.
3. $\mathcal{A}$ obtain a signature $\sigma_i^{(h)}$ on the message $m^{(h)}$ in the structure $\psi_i^{(h)}$.

After at most q_s iterations from step 1 to step 3, $\mathcal{A}$ outputs a forgery on a message m^* in the signer structure ψ_i^*, where it holds $(m^*, \psi_i^*) \notin \{(m^{(h)}, \psi_i^{(h)})\}_{h=1}^{q_s}$ and **Verification** algorithm outputs *accept* on m^* in ψ_i^*.

Definition 8. An adversary $\mathcal{A}$ breaks a structured multisignature scheme with $(t, q_s, Q_F, Q_H, n, \epsilon)$ where Q_H means $q_{F_1}, \cdots, q_{F_n}$ and Q_H means q_{H_1}, $\cdots, q_{H_n}$ if A which does not know sk_i can output a forgery that meets the conditions described above with a success probability greater than ϵ within an execution time t in the above game. Here, $\mathcal{A}$ can send a query at most q_s times, q_{F_i} query to F_i and q_{H_i} query to H_i. A structured multisignature is $(t, q_s,\ Q_f, Q_h, \epsilon)$-secure if there is no adversary who can break a structured multisignature scheme with $(t,\ q_s, Q_f, Q_h, \epsilon)$.

4.3.4 Other Attacks

In this section, we describe two attacks, rogue key attack [67] and attack-0 [90].

Rogue Key Attack According to Micali et al.[67], DLP-based schemes such as the scheme in Section 3.2 of the paper [67] may possess weakness related to key setup. An adversary who is a member of a signing group generates a public key as a function of other signers' public keys. For example, let the number of signers be n, a public parameter be (p, g), each signer's secret key be x_i and its public key be $y_i = g^{x_i} \bmod p$, and adversary's secret key be x_a. Then, the adversary computes the following equation as a public key y_a.

$$y_a = \left(\prod_{i=1}^{n-1} y_i\right)^{-1} \cdot g^{x_a} (\mathrm{mod} p). \tag{4.7}$$

In this case, the adversary can easily forge a multisignature for the group of signers, because x_a ends up being the group's secret key corresponding a public key $\prod_{i=1}^{n} y_i$ of the whole group.

Attack-0 In order to prevent this attack, the number of signers in [90] is restricted. In their scheme, the verification equation is as follows:

$$g^{X_i} = \prod_{i \in \psi_n} \left(R_i^{e_i} y_i^{d_i} \right)^{w_i} (\mathrm{mod} p), \tag{4.8}$$

where (p, g) mean the public parameter, X_i and R_i mean the signature value, y_i means each signer's public key, e_i and d_i mean hash values, and w_i means a weight of the graph. The weight w_i of the graph is the number of path passing through ID_i in the whole structure ψ_n. Hence, colluding signers may be able to construct a graph such that w_i is zero modulo q. Then for any message m and for any structure ψ_n, the colluding signers may easily cancel out the target signer's signature. To avoid this attack, it must hold $w_i < q$ for any ψ_n. From Lemma 2.3 in [90], the weight of graph consisting of n signers is $w_i \leq 3^{n/3}$ for any graph ψ_n. Hence, $n < \frac{3}{\log 3} \log q$ must hold to prevent this attack. Thus, the number of signers is restricted in [90].

4.4 Multi Round Identification Schemes

In this section, we describe a syntax of multi round identification scheme and its security model. These are used for analyzing the security of the proposed scheme. We note that **Multi Round Identification** algorithm described below is an interactive algorithm between a prover and a verifier: in particular, the prover executes steps for generating random numbers and

4.4.1 Syntax

Setup Given a security parameter 1^k as input, output a public parameter $para$.

Key Generation Given $para$ as input, generate n pairs of secret key sk_i and its public key pk_i.

Multi Round Identification Given a signer structure ψ_n generated by n signers and a random string m as message, repeat the following steps for $i = 1$ to n.

1. Extract a subgraph ψ_i from ψ_n and generates random number R_i and then generate a signature σ_j for all j such that $\psi_j \subset \psi_i$.
2. Generates random numbers (d_i, e_i), where d_i and e_i are responses of random oracle queries h_i and f_i respectively.
3. Generate $\sigma_i \leftarrow F(m, e_i, d_i, \{X_j\}_{j \in \mathcal{T}(i)}, \psi_i, sk_i)$.

After these iterations, Check if this multi-round identification is valid.

4.4.2 Attack Model for Multi Round Identification Scheme

Security of multi round identification scheme is defined as follows. The process of the game is almost the same as that of the structured multisignature scheme, and hence we omit the detail. The goal of the game is that, given system parameter *para* and all signers' public keys pk_i, an adversary A executes the following security game with an honest verifier V. Then, A passes to the verification for any $\psi_n \in \mathcal{G}$.

4.4.3 Definition of the Security

Definition 9. Adversary A breaks a multi round identification scheme with (t, ϵ) if A who does not know a secret key x_i of a prover can pass to the security game with an honest verifier V for any signer structure ψ_n consisting of n signers with the success probability greater than ϵ within the execution time t. Here, A is a passive attacker which does not act as a verifier with true prover.

Definition 10. A multi round identification scheme is (t, ϵ)-secure if there is no adversary who can break the scheme with (t, ϵ).

4.5 Proposed Scheme

In this section, we propose a structured multisignature scheme. First, we explain the proposed structured multisignature scheme, then explain the multi round identification scheme.

4.5.1 Structured Multisignature Scheme

In this scheme, each signer pk_i has hash functions $H_i : \{0,1\}^* \to \{0,1\}^q, F_i : \{0,1\}^* \to \{0,1\}^q$. We assume an existence of a trusted center that generates system parameters. Without loss of generality, we assume that each

multisignature is processed by pk_i in the order from signer ID_1 to ID_n. We also assume each signer appears only once in a signer structure, so that each signer executes the signature generation only once for the generation of one multisignature.

Setup Given a security parameter 1^k, generate a large prime number p and q such that $q|(p-1)$, and choose an element g of order q from $\mathbb{Z}_p^*$ and pseudo-random function (PRF) keys K_1, K_2. Then, return $(\mathbb{Z}_p^*, g, q, K_1, K_2)$ as a public parameter $para$ where $PRF : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_q^*$ be a pseudo-random function family.

Key Generation Given $para$, generate a random number $x_i \in \mathbb{Z}_q$ as sk_i. Then compute $y_i = g^{x_i} \bmod p$, and choose hash functions $H_i : \{0,1\}^* \to \mathbb{Z}_q^*$ and $F_i : \{0,1\}^* \to \mathbb{Z}_q^*$ as pk_i. Return (x_i, y_i, H_i, F_i).

Signing Given a message m, previous signers' signatures σ_j and structures ψ_j for $j \in \mathcal{T}(i)$ and sk_i, execute the following steps.

Check if ψ_i consists of only ith signer. If not, execute the verification of each σ_j by using the verification algorithm, and abort the process if there is invalid signature for σ_j. If this signer is the first signer, i.e., $\psi_{i-1} =$, then the verification process is skipped.

Then, compose $\psi_{\mathcal{T}(i)} := \cup_{j \in \mathcal{T}(i)} \psi_j$ as the composition structure of ψ_j for $j \in \mathcal{T}(i)$. Then, compose $\psi_i := \psi_{\mathcal{T}(i)} \cap \phi_i$. If this signer is the first one, set ϕ_i to ψ_i. Then, choose a random number $r_{i,2} \leftarrow \mathbb{Z}_q$ and compute $R_{i,2} = g^{r_{i,2}} \bmod p$. Set $R_{i,1} = PRF_{K_1}(m, R_{i,2}, \psi_i)$ and $R_{i,3} = PRF_{K_2}(m, R_{i,2}, \psi_i)$, and then compute as follows:

$$S' \equiv \begin{pmatrix} a_{i,1} & e_i x_i + d_i r_{i,2} \bmod q \\ 0 & a_{i,3} \end{pmatrix} \equiv \begin{pmatrix} a_{i,1} & v_i \\ 0 & a_{i,3} \end{pmatrix}, \tag{4.9}$$

where $a_{i,1} = f_i(m, R_{i,1}, \psi_i), a_{i,3} = h_i(m, R_{i,3}, \psi_i)$, $e_i = f_i(m, R_{1,2} \parallel R_{j,2}, \psi_i)$ and $d_i = h_i(m, R_{1,2} \parallel R_{j,2}, \psi_i)$. Then, compute $S_i = (\sum_{\dot{+}, k \in \mathcal{T}(i)} S_k) \odot \S'$ where $\sum_{\dot{+}, k \in \mathcal{T}(i)}$ denotes the summation by $\dot{+}$ for $k \in \mathcal{T}(i)$. If this signer is the first one, set $S_i = S'$. Set $\mathbb{R}_{i,j} = (R_{1,j}, R_{2,j}, \cdots, R_{i,j})$ for $j \in [1,3]$. Then, output $(S_i, \{\mathbb{R}_{i,j}\}_{j \in [1,3]})$ as σ_i. If there are j signers at the terminal for ψ_i, then the whole signature σ_i consists of $S_i = \sum_{\dot{+}, k=1}^{j} S_k$ and $\mathbb{R}_{i,j} = (R_{1,j}, R_{2,j}, \cdots, R_{i,j})$ for $j \in [1,3]$.

Verification Check if the following congruence holds:

$$\begin{aligned} F(S_n) \equiv & \begin{pmatrix} a_{1,1} & y_1^{e_1} \cdot R_{1,2}^{d_1} \\ 0 & a_{1,3} \end{pmatrix} \oplus_1 \begin{pmatrix} a_{2,1} & y_2^{e_2} \cdot R_{2,2}^{d_2} \\ 0 & a_{2,3} \end{pmatrix} \oplus_2 \cdots \\ & \oplus_{n-1} \begin{pmatrix} a_{n,1} & y_n^{e_n} \cdot R_{n,2}^{d_n} \\ 0 & a_{n,3} \end{pmatrix}, \end{aligned} \tag{4.10}$$

where $\oplus_i$ means the addition $\dagger$ if the relation between ith signer and $i+1$th signer is a parallel structure, otherwise, the multiplication $\circ$. In addition, for the calculation of the right hand side of the congruence, $\dagger$ operation is searched and any found $\dagger$ operation is computed at first one by one until all $\dagger$ operations are computed. Then all $\circ$ operations are computed one by one from the leftmost $\circ$ operation until the rightmost one.

4.5.2 Multi Round Identification Scheme

To analyze the proposed scheme, we consider two multi round identification schemes as follows. The first one considers identifications of all signers for all signing structures. The second one considers identifications of signers for a fixed signer structure. There are two entities, a prover P and a verifier V, and **Multi Round Identification** is an interactive algorithm among them. We also assume an existence of a trusted center to generate a public parameter $para$.

Scheme-A

The following scheme is a multi round identification scheme for all signing structures.

Setup Given a security parameter 1^k, generate a big prime number p and q such that $q|(p-1)$. Then, choose an element g of an order q from $\mathbb{Z}_p^*$. Output $(\mathbb{Z}_p^*, g, q)$ as $para$.

Key Generation Given $para$, generate n pairs of a secret key $x_i \in \mathbb{Z}_q$ and a public key y_i such that $y_i = g^{x_i} \bmod p$, where n is the number of signers. Return $\{y_i\}_{i=1,\cdots,n}$.

Multi Round Identification Given ψ_n consisting of n signers and a random string m as a message, repeat the following step from $i = 1$ to n.

1. Extract ψ_i from ψ_n and generate a random number $r_{i,2}$. Then, compute $R_{i,2} = g^{r_{i,2}}$, $R_{i,1} = PRF_{K_1}(m, R_{i,2}, \psi_i)$ and $R_{i,3} = PRF_{K_2}(m, R_{i,2}, \psi_i)$.

2. Generate random numbers $a_{i,1}, a_{i,3}, d_i$ and e_i, where $a_{i,1}, a_{i,3}, d_i$ and e_i are responses of random oracle queries h_i and f_i respectively.

3. Compute as follows:

$$S' \equiv \begin{pmatrix} a_{i,1} & e_i x_i + d_i r_{i,2} \bmod q \\ 0 & a_{i,3} \end{pmatrix} \equiv \begin{pmatrix} a_{i,1} & v_i \\ 0 & a_{i,3} \end{pmatrix}, \quad (4.11)$$

   Compute $S_i = (\sum_{\dot{+}, j \in \mathcal{T}(i)} S_j) \odot S'$ where $\sum_{\dot{+}, j \in \mathcal{T}(i)}$ means the summation of $\dot{+}$ for $j \in \mathcal{T}(i)$.

After these iterations, for $(m, \{\mathbb{R}_{n,j}\}_{j \in [1,3]}, S_n, \psi_n)$ where $\mathbb{R}_{n,j} = (u_{1,j}, \cdots, u_{n,j})$, check if the following equation holds:

$$F(S_n) \equiv \begin{pmatrix} a_{1,1} & y_1^{e_1} \cdot u_{1,2}^{d_1} \\ 0 & a_{1,3} \end{pmatrix} \oplus_1 \begin{pmatrix} a_{2,1} & y_2^{e_2} \cdot u_{2,2}^{d_2} \\ 0 & a_{2,3} \end{pmatrix} \oplus_2 \cdots \oplus_{n-1} \begin{pmatrix} a_{n,1} & y_n^{e_n} \cdot u_{n,2}^{d_n} \\ 0 & a_{n,3} \end{pmatrix},$$

where the rule for the operation $\oplus_i$ is the same as described for the congruence (4.11).

Scheme-B

The following scheme is a multi round identification scheme with a fixed signing structure, and the signing structure is declared before **Multi Round Identification** algorithm.

Setup This step is the same as that in Scheme-A.

Key Generation This step is the same as that in Scheme-A.

Structure Declaration P chooses a signer structure ψ_n with n signers and publishes ψ_n.

Multi-Round Identification This step is the same as that in Scheme-A.

4.5.3 Reduction Lemma

In this subsection, we recall the reduction lemma [73]. We reduce the security of the structured multisignature scheme into the security of the multi round identification scheme.

Definition 11. We say that a multi round identification scheme has the perfect zero-knowledge property, if there is a polynomial-time machine $\mathcal{S}$ with input all public key y_i and any ψ_n consisting of n signers, which satisfies the following equation:

$$\sum_{\alpha,\beta,\gamma,\delta} \left| \begin{array}{l} \Pr\left[\begin{array}{c} (\mathbb{R}_{n,j}, \mathbb{E}_n, \mathbb{D}_n, S_n) \leftarrow [P(\mathbb{X}_n, \psi_n), V(\mathbb{Y}_n, \psi_n)] : \\ (\mathbb{R}_{n,j}, \mathbb{E}_n, \mathbb{D}_n, S_n) = (\alpha, \beta, \gamma, \delta) \end{array} \right] \\ \quad - \Pr\left[\begin{array}{c} (\mathbb{R}'_{n,j}, \mathbb{E}'_n, \mathbb{D}'_n, S'_n) \leftarrow \mathcal{S}(\mathbb{Y}_n, \psi_n) : \\ (\mathbb{R}'_{n,j}, \mathbb{E}'_n, \mathbb{D}'_n, S'_n) = (\alpha, \beta, \gamma, \delta) \end{array} \right] \end{array} \right| = 0,$$

where $(\mathbb{R}_{n,j}, \mathbb{E}_n, \mathbb{D}_n, S_n) \leftarrow [P(\mathbb{X}_n, \psi_n), V(\mathbb{Y}_n, \psi_n)]$ means $(\mathbb{R}_{n,j}, \mathbb{E}_n, \mathbb{D}_n, S_n)$ is obtained by the execution of Scheme-B between $P(\mathbb{X}_n, \psi_n)$ and $V(\mathbb{Y}_n, \psi_n)$, $\mathbb{E}_n$ means $\{e_i\}_{i=1,\cdots,n}$, $\mathbb{D}_n$ means $\{d_i\}_{i=1,\cdots,n}$, $\mathbb{X}_n$ means $\{x_i\}_{i=1,\cdots,n}$ and $\mathbb{Y}_n$ means $\{y_i\}_{i=1,\cdots,n}$.

Lemma 12. Scheme-B has the perfect zero-knowledge property.

Proof. (Sketch) The proof is almost the same as that in [90]. □

Since Scheme-B has the perfect zero-knowledge property, we can obtain the following ID-reduction lemma similarly with the papers [73, 90].

Lemma 13 (Reduction Lemma)**.** Let $\epsilon_1 \geq \frac{1}{q}(q_{H_1}(q_{F_1}(q_{H_2}(\cdots(q_{F_n}(\frac{2^{n+1}}{q^{n-1}} + q_s) + 1)\cdots) + 1) + 1) + 1)$.

1. If A_1 breaks the proposed scheme with $(t_1, q_s, Q_F, Q_H, \epsilon_1)$, then there exists A_2 who breaks the scheme with $(t_1, q_s, \mathbf{1}, \mathbf{1}, \epsilon_2)$, where $\epsilon_2 = \epsilon_{H_n}$ and $\mathbf{1}$ means n-tuple $(1, \cdots, 1)$. Here, let $\epsilon_{H_0} = \epsilon_1, \epsilon_{F_i} = \frac{\epsilon_{H_{i-1}} - \frac{1}{q}}{q_{F_i}}, \epsilon_{H_i} = \frac{\epsilon_{F_i} - \frac{1}{q}}{q_{H_i}}$.
2. If A_2 breaks the proposed scheme with $(t_1, q_s, \mathbf{1}, \mathbf{1}, \epsilon_2)$, there exists A_3 who breaks the scheme with $(t_3, 0, \mathbf{1}, \mathbf{1}, \epsilon_3)$, where $t_3 = t + \mathcal{O}(q_s)$ and $\epsilon_3 \geq \epsilon_2 - \frac{q_s}{q}$.
3. If A_3 breaks the proposed scheme with $(t_3, 0, \mathbf{1}, \mathbf{1}, \epsilon_3)$, there exists A_4 who breaks a multi-round identification scheme corresponding to the proposed scheme with (t_3, ϵ_3).

Proof. (Sketch) The proof is almost the same as Lemma 4.6 in the paper [90]. □

4.5.4 Hierarchical Heavy Row Lemma

We adopt the ideas of the hierarchical structures of a boolean matrix and heavy row introduced in [73] in order to prove the security of our scheme. We assume that there is a cheater A who can break a multi round identification scheme with (t, ϵ).

Definition 14. The possible outcomes of the execution of a cheater A and an honest verifier V are denoted by a boolean matrix $\mathcal{H}_i(r, e_1, d_1, \cdots, e_{i-1}, d_{i-1}; e_i, d_i, \cdots, e_n, d_n)$. The rows of the matrix correspond to all possible choices of $r, e_1, d_1, \cdots, e_{i-1}, d_{i-1}$, where r is a random tape and $e_1, d_1, \cdots, e_{i-1}, d_{i-1}$ are outputs of hash functions. The columns of the matrix correspond to all possible choices of $e_i, d_i, \cdots, e_n, d_n$. Its entries are 0 if V rejects A's proof, and 1 if V accepts A's proof. A slightly different boolean matrix with an upper script + is defined by $\mathcal{H}_i^+(r, e_1, d_1, \cdots, e_{i-1}, d_{i-1}, e_i; d_i, \cdots, e_n, d_n)$, where the difference is the place of e_i. Finally, a boolean matrix without any upper or lower script is defined by $\mathcal{H}(r, e_1, d_1, \cdots, e_{i-1}, d_{i-1}, e_i, d_i, \cdots, e_n, d_n;)$, where all of $r, e_1, d_1, \cdots, e_{i-1}, d_{i-1}, e_i, d_i, \cdots, e_n, d_n$ are placed on the left of the ";" mark.

Definition 15. A row of matrix $\mathcal{H}_i$ is i-heavy if the fraction of 1's along the row is at least $\epsilon/2^i$, where ϵ is the success probability of A. A row of matrix $\mathcal{H}_i^+$ is i-heavy if the fraction of 1's along the row is at least $\epsilon/2^{i+1}$.

Lemma 16 (Hierarchical Heavy Row Lemma)**.** If the 1's in $\mathcal{H}$ are located in 1-heavy rows of $\mathcal{H}_1$, 2-heavy rows of $\mathcal{H}_2, \cdots, (i-1)$-heavy rows of $\mathcal{H}_{i-1}$ simultaneously, then they are also located in i-heavy rows of $\mathcal{H}_i$ and $\mathcal{H}_i^+$ with a probability of at least $\frac{1}{2}$.

Proof. The proof is shown in Lemma 12 in [73]. □

4.5.5 Security of the Proposed Scheme

First, we show the security of the multi round identification scheme. The following lemma holds for Scheme-A, because of the definition of multi round identification adversary.

Lemma 17. If there is an adversary who can break the multi round identification scheme with (t, ϵ), then there is an algorithm which can solve the

DLP problem at n times with $(t(n), \epsilon(n))$ where

$$t(n) = (t + t_{mri})\frac{2^{(2n+1)} + 3 \cdot 2^{n+1}(n-1) + 1}{3\epsilon}, \tag{4.12}$$

$$\epsilon(n) = \left(1 - \frac{1}{q}\right)\left(\frac{1}{2}\right)^{2^n+n-2} p_0(\epsilon)\left(\prod_{i=1}^{n} p_i(\epsilon)^{2^{i-1}}\right)(p_{n+1}(\epsilon))^{n-1}, \tag{4.13}$$

t_{mri} is the execution time of the identification scheme, and

$$p_0(\epsilon) = \left(1 - (1-\epsilon)^{\frac{1}{\epsilon}}\right), \tag{4.14}$$

$$p_i(\epsilon) = \left(1 - \left(1 - \frac{\epsilon}{2^i}\right)^{\frac{2^i}{\epsilon}}\right). \tag{4.15}$$

Proof. Assume that there is an adversary $\mathcal{A}_4$ which can break a multi round identification scheme with (t, ϵ). Then, we construct an algorithm $\mathcal{B}$ which can break the DLP assumption by using $\mathcal{A}_4$. From the definition of boolean matrix (Definition 14), the possible outcomes of the execution of $\mathcal{A}_4$ and an honest verifier V are denoted by boolean matrix $\mathcal{H}_i(r, e_1, d_1, \cdots, e_{i-1}, d_{i-1}; e_i, d_i, \cdots, e_n, d_n)$. To find each secret key, we take the following strategy.

1. Probe random entries in $\mathcal{H}$ to find an entry $a^{(0)}$ with 1.

2. We denote by $\mathcal{H}_1^{(0)}$ the rows where $a^{(0)}$ is located in $\mathcal{H}_1$. After $a^{(0)}$ is found, probe random entries in $\mathcal{H}_1^{(0)}$ to find another entry $a^{(1)}$ with 1.

3. This step is repeated for $j \in [2, n]$. We denote by $\mathcal{H}_j^{(i)}$ the row of $\mathcal{H}_j$ where $a^{(i)}$ is located for each $i \in [0, n(j-1)-1]$, where for $k \in [0, n]$ the function $n(k)$ denotes the total number of entries which have already been found until the execution of this step is finished. After $a^{(0)}, \cdots, a^{n(j-1)-1)}$ are found, probe random entries to find another entry $a^{(n(j-1)-1+i)}$ with 1 for $i \in [1, n(j-1)]$.

4. We denote by $\mathcal{H}_n^+$ the row of $\mathcal{H}_n^+$ where $a^{(i)}$ is located for each $i \in [0, n-2]$. After the kth step for $k \in [0, n]$, probe random entries in $\mathcal{H}_n^{+(i)}$ to find another entries $a'^{(i)}$ with 1 for $i \in [0, n-2]$.

After the above steps, we can obtain $2^n + n - 1$ simultaneous equations.

Here, in the simultaneous equations, we have 2^n unknowns which consist of n secret keys $\{x_i\}_{i=1}^n$ and n random numbers $\{r_{i,2}\}_{i=1}^n$. In addition, we

have more unknowns by the execution of the above steps. Therefore, the total number of unknowns is 2^n+n-1. Since the number of the unknowns is equal to the number of equations, $\mathcal{B}$ can find the value of x_i and r_i with the probability of $1-\frac{1}{q}$ which is the probability that the simultaneous equations are regular. The success probability $\epsilon(n)$ and the calculation time $t(n)$ of $\mathcal{B}$ can be written by the definition of the adversary, Definition 15 and Lemma 16 as follows. In the above probing step,

1. Find the entry by probing in $\mathcal{H}$ $1/\epsilon$ times.
2. Find the entry by probing in $\mathcal{H}_1$ $2/\epsilon$ times.
3. For $j \in [2, n]$, find the entry by probing in $\mathcal{H}_k$ $2^k/\epsilon$ times.
4. For each $i \in [0, n-2]$, find the entry by probing in $\mathcal{H}_n^+$ $2^{n+1}/\epsilon$ times.

Therefore, the execution time can be written as follows:

$$\begin{aligned} t(n) &= \frac{t+t_{mri}}{\epsilon}\left(1+\sum_{j=1}^{n} 2^{2k-1}+2^{n+1}(n-1)\right) \\ &= (t+t_{mri})\frac{2^{(2n+1)}+3\cdot 2^{n+1}(n-1)+1}{3\epsilon}, \end{aligned}$$

where t_{mri} means the execution time of the multi round identification scheme. Similarly, the success probability can be written as follows:

$$\begin{aligned} \epsilon(n) &= \left(1-\frac{1}{q}\right) p_0(\epsilon)\left(\prod_{j=1}^{n}(\frac{1}{2}p_j(\epsilon))^{2^{j-1}}\right)\left(\frac{1}{2}p_{n+1}(\epsilon)\right)^{n-1} \\ &= \left(1-\frac{1}{q}\right)\left(\frac{1}{2}\right)^{2^n+n-2} p_0(\epsilon)\left(\prod_{i=1}^{n} p_i(\epsilon)^{2^{i-1}}\right)(p_{n+1}(\epsilon))^{n-1}, \end{aligned}$$

where, for all i, the following equation holds:

$$p_i(\epsilon) = \left(1-\left(1-\frac{\epsilon}{2^i}\right)^{\frac{2^i}{\epsilon}}\right).$$

□

The following theorem can be obtained from the above lemma and the reduction lemma.

Theorem 18. If there is an adversary who can break the proposed structured multisignature scheme with $(t, q_s, Q_F, Q_H, \epsilon)$, then there is an algorithm which can solve the DLP at n times with $(t(n), \epsilon(n))$, where

$$\begin{aligned} t(n) &= (t + t_{mri} + t_s)\frac{2^{(2n+1)} + 3 \cdot 2^{n+1}(n-1) + 1}{3\epsilon_3}, \\ \epsilon(n) &= \left(1 - \frac{1}{q}\right)\left(\frac{1}{2}\right)^{2^n+n-2} p_0(\epsilon)\left(\prod_{i=1}^{n} p_i(\epsilon_3)^{2^{i-1}}\right)(p_{n+1}(\epsilon_3))^{n-1}, \end{aligned}$$

t_{mri} is the execution time of the multi round identification scheme, t_s is the simulation time of q_s signatures, and

$$\begin{aligned} p_0(\epsilon) &= \left(1 - (1-\epsilon_3)^{\frac{1}{\epsilon_3}}\right), \\ p_i(\epsilon) &= \left(1 - \left(1 - \frac{\epsilon_3}{2^i}\right)^{\frac{2^i}{\epsilon_3}}\right). \end{aligned}$$

Discussion about Known Attacks

In this subsection, we discuss the security about known attacks described in section 4.3.4.

Rogue Key Attack To prevent this attack, Bellare and Neven proposed a simple solution such that the exponent of each public key is always different by using hash function, except with negligible probability in [10]. We have adopted this idea to the proposed scheme, and so our scheme can be prevented the effect of attacking key generation.

Attack-0 As explained in 4.3.4, colluding signers may be able to construct a graph such that the weight w_i of the graph is zero mod q so that the target signer's signature is cancelled out in the verification equation (8). In contrast, the verification congruence (11) of our scheme does not include the weight w_i of graph. This property can be obtained by the operation on the ring homomorphism by Chida et al. [26]. Moreover, multisignatures in our scheme are computed by $S_i = (\sum_{\dot{+},j\in\mathcal{T}(i)} S_j) \odot S'$. For the multiplication $\odot$, the (1,1)-element of $\sum_{\dot{+},j\in\mathcal{T}(i)} S_j$ and the (2,2)-element of S', which are random numbers, are multiplied to the (1,2)-elements of $\sum_{\dot{+},j\in\mathcal{T}(i)} S_j$ and S', respectively. As a result, the (1,2)-element of the matrix in the right hand of the verification congruence (11) can be written as $\prod_i \left(y_i^{e_i} \cdot u_{i,2}^{d_i}\right)^{\sum_\alpha A_\alpha}$,

Table 4.1: Performance evaluation for n signers

	Security Assumption	Order Flexibility	Signer Structure	Restriction in the number of signers
BDD+00 [19]	(DLP)	No	Serial, Parallel	No
LZL04 [61]	CDH	No	Serial, Parallel	No
MM00 [68]	(DLP)	Yes	Serial	Yes
OCSN96 [72]	(DLP)	Yes	Serial	Yes
Tada03 [90]	DLP	Yes	Serial, Parallel	No
WOMOD07 [94]	CDH	Yes	Serial, Parallel	Yes
Our scheme	DLP	Yes	Serial, Parallel	Yes

where A_α is a term $\prod_\beta a_{\beta,j}$ for $j \in \{1, 3\}$, β selected in this product is determined by the given ψ_n and $\beta \neq i$. Although A_α may be determined only by colluding signers' variables $a_{\beta,j}$, each $a_{\beta,j}$ in the product is a hash value. Since hash functions F_i and H_i for computing $a_{\beta,l}$ are collision resistant, the probability that adversaries find appropriate hash values that cancel out the signature of target signer is negligible if q is enough large. In other words, the probability that $\sum_\alpha A_\alpha$ becomes zero mod q is negligible without the restriction on the number of signers.

4.6 Evaluation of the Proposed Scheme

We compare the proposed scheme with DLP-based schemes in [19, 61, 68, 72, 90, 94] with respect to security assumption, order flexibility, signing structure and restriction of the number of signers. We showed the result as Table.2. Concerning the security assumption, the BDD+00 scheme [19], the MM00 scheme [68], the OCSN96 scheme [72], the Tada03 scheme [90] and our scheme are based on the DLP assumption, and only the Tada03 scheme and our scheme are proven as difficult as solving the DLP (unproven schemes are shown with (DLP)). We discuss about the property among them. As shown in Table 4.2, the BDD+00 scheme, the LZL04 scheme, the OCSN96 scheme, the Tada03 scheme, the WOMOD07 scheme and our scheme deal

with both the serial structures and the parallel structures. In contrast with the Tada03 scheme, our proposed scheme has no restriction in the number of signers since our scheme can prevent the attack-0 without restricting the number of signers.

Chapter 5

BGP-Aiding Aggregate Signatures

5.1 Introduction

5.1.1 Motivation

Overview A problem for designing the Internet is injections of false information in routing information of autonomous systems (ASes) in border gateway protocol (BGP) [79]. For instance, in 2008, the incident called YouTube Hijacking [80] occurred where a path information to Youtube [104] was mistakenly replaced with that to Pakistan Telecom. One of approaches to resist these threats is to guarantee the validity of the path information, and *secure-border gateway protocol (S-BGP)* [52] or *border gateway protocol security extension (BGPSEC)* [59] have been focused as such technologies. However, there are two problems preventing development of the technology. One of the problems is overloads for routers [89], and the routers require a large amount of memory such as several tens of giga bytes. Another problem is a packet limitation of BGP. The limitation is 4096 bytes [79], and the total size of the packets including signatures and informations of public keys has to be smaller than the limitation. Nevertheless, there are 65,535 AS numbers (ASNs) [46] and their public key information. Hence BGP security from the existing technologies is impossible since the total size of the public keys are linear. In this paper, in order to address these problems, we design a practical and provably-secure scheme for securing BGP.

An *aggregate signatures scheme* [17] is a cryptographic approach against this problem, and the scheme allows n signers to generate n signatures on

n individual messages and to combine all of these signatures into a single short signature. For instance, by utilizing aggregate signatures, each router signs its own routing table and aggregates their signatures with ones given from other routers into a short signature. Hence, many researchers have alleged that a key application of aggregate signatures is for BGP security. However, we point out problems of these schemes in the interoperation of BGP in two points of view.

The first point is a gap between recent development of aggregate signatures and a realistic setting of BGP security. More specifically, as indicated in the latest researches [15, 18, 35, 37, 54, 55], the state-of-the-art aggregate signature schemes are intended for sequential aggregate signatures [66] with aggregate-signing where each signer aggregates signatures at the same time as signing. Meanwhile, recent BGP technologies are intended for multipath [4, 88]. That is, there are multiple paths between any routers in order for improvements of the availability and the throughput. Intuitively, this means that sequential aggregate signatures disallow to aggregate signatures between paths, and an individual signature is generated for each path. Unfortunately, developments of aggregate signatures in recent years is inapplicable for the key application. Here, the gap occur in the case of multipath BGP which is the latest technology for BGP. We note that the sequential aggregate signatures are applicable for the traditional BGP security [52] and that these signatures give rise to many practical applications such as certificate chain.

The second problem is the efficiency. BGP security apparently becomes practical by aggregate signatures. However, according to a specification of BGP security extension [58], subject key identifiers (SKIs), which are unique identifiers of public keys, are included in a BGP packet. Hence, the packet limitation is still a major problem even if the aggregate signatures are available.

5.1.2 Our Contribution

In this chapter, we discuss an aggregate signature scheme which is the best practice in routing security and call such a scheme *BGP-aiding aggregate signature* scheme. In particular, contributions of this work is as follows: (1) we describe requirements of BGP-aiding aggregate signature scheme; (2) we propose the modified-Gentry-Ramzan scheme.

Requirements of a BGP-aiding aggregate signature scheme are as follows: (a) the size of signatures is fixed with respect to the number of signers; (b) any third party can compress individual signatures into a single short signature, and (c) the scheme is an ID-based scheme. For the first reason,

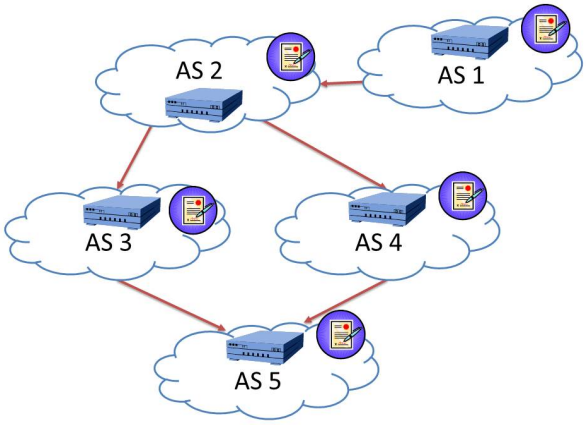


Figure 5.1: BGP Multihoming

BGP has a restriction in the packet size, which is 4,096 bytes, and hence the size of signatures has to be independent of the number of signers. For the second reason, sequential aggregate signatures, which are a mainstream in recent schemes, allows only signers who have secret keys to compress signatures, and hence an individual signature is generated for each path in the multi-path setting. This property brings down the improvement of the efficiency by aggregate signatures. In the light of a specification of BGP, an aggregation of signatures without secret keys, called *general aggregation*, is crucial. The general aggregation essentially support the multi-path. Finally we describe the third reason. In the specification of BGP, packets include subject key identifier to identify public keys utilized to sign. Even if the size of signatures is fixed, we have to consider the restriction in the packet size since the the number of the subject key identifier becomes linear. Cryptographic scheme suitable in such an environment as conventional framework is ID-based cryptosystem. In ID-based cryptosystem, each user can utilize any string as a public key, and hence the user can easily manage a certificate. We note that ID-based signatures have high affinity with BGP security in the sense that certificates become unnecessary.

In this work, we modify the scheme by Gentry and Ramzan [36], called GR06 scheme for short, to fit into BGP. In the existing schemes, there are several schemes which have a similar capability as the requirements of the BGP-aiding aggregate signature scheme, but trivially utilizing these schemes may have a vulnerability. In particular, there is a potential vulnerability against by increasing the number of signers, and hence we have to consider

a restriction in the number of the signers. Only the modified-GR06 scheme is available in such a situation.

5.1.3 Related Work

In this section, we describe several works in terms of aggregate signatures and combinations of digital signatures and BGP security.

Aggregate signatures [17] are a general variant of multisignatures [48]. The most famous one in this framework was proposed by Boneh et al. [17] while an older scheme with such a message-flexibility was proposed by Mitomi and Miyaji [68] individually. There are three types of aggregate signatures, i.e., general one [17], sequential one [66] and synchronized one [36]. The first scheme of the general type was proposed by Boneh et al., and this type requires an interactive process. After, the first sequential scheme was proposed by Lysyanskaya et al. [66], and this type allows each signer to aggregate by sequentially passing the aggregate from one signer to the next. In the sequential type, the signers can execute signing generations and their aggregation at the same time, and hence the traffic in the whole network can be reduced. Synchronized aggregate signatures were defined by Gentry and Ramzan [36], and Ahn et al. [1] gave a more detail definition. The synchronized type allows signers to share state-informations, and signatures with the same state-information can be aggregated. The scheme discussed in this paper has the properties of all the three schemes.

There are several works with respect to BGP security introducing digital signature schemes. Zhao et al. designed a system combining an aggregate signature scheme with BGP, and evaluated

5.2 BGP and Its Security Extension

In this section, we describe BGP and S-BGP as its security extension.

5.2.1 Border Gateway Protocol

Overview

Border gateway protocol is a path vector routing protocol. The current Internet is a large network that links together smaller networks to each other, and each network is called an autonomous system (AS). In recent, there are 36,000 ASes in the Internet [25]. Each AS has a unique number represented by a unique 2-byte as an AS number (ASN), and manages its

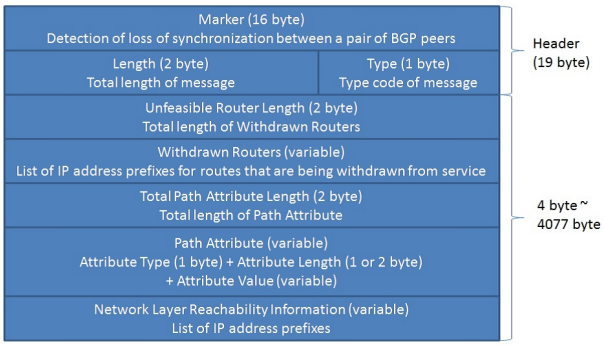


Figure 5.2: Update Message of BGP

subnetworks described as IP prefixes which are ranges of IP addresses. ASNs are assigned in blocks by Internet Assigned Numbers Authority (IANA) to Regional Internet Registries (RIRs). Recursively, RIRs assign ASNs from IANA within their designated areas.

More specifically, routers executing BGP are called BGP speakers. The BGP speakers maintain connections called BGP sessions with neighboring speakers, and send an update message to advertise a new preferred route to its prefix. The message consists of attribute information, and the important ones are IP prefix and AS path. An AS path is a sequence of AS numbers that specifies a sequence of ASes in the network, and especially the last AS in the sequence is called the originator of this route. The BGP speakers keep routes in routing information basis (RIB). One Adj-RIBs-In per keeps received routes from the peer, and Loc-RIB records all preferred routes for each prefix. Hereafter, we define the Adj-RIBs-In and Loc-RIB as a routing table for the BGP speaker. Ordinarily, when each speaker adds a new route, a Loc RIB of the speaker replaces a previously preferred route with the new one.

We give the detail of a packet in BGP in Fig.5.2.1. The limitation of the packet is 4096 bytes, where the size of the header is 19 bytes, the length information of the path is four bytes and the rest of 4,073 bytes is variable space for the path information.

Multi-Path Routing

In recent BGP, the multi-path is adopted from following viewpoints. The Detail of advantages of the multi-path has been described in [91].

Higher Network Capacity By utilizing the multi-path, it is possible to

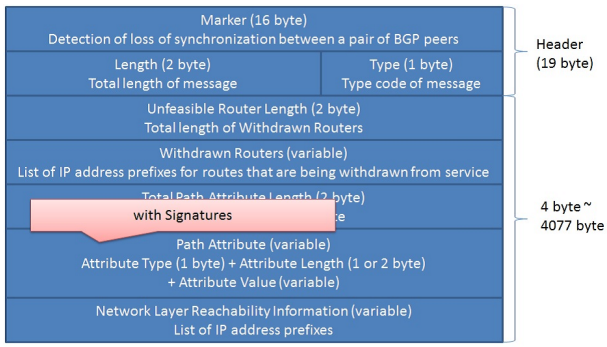


Figure 5.3: Update Message of BGPSEC

push more traffic through the network, because the traffic flow through any path that has available capacity.

Improved Response to Path Changes If any path becomes failed, BGP reroutes packets through alternate routes. However, its convergence time in the single path setting may be measured in minutes, and this means that a part of the network becomes unavailable. The convergence time should be improved as fast as possible, and the multi-path setting has a potential advantage to improve the response time to seconds [29].

Enhanced Security Man-in-the-middle attacks are much harder to achieve, because a single interception point is insufficient and an attacker needs to be located along the multiple paths. In addition, even if denial-of-service attacks occurs, the multi-path setting allows operators to move the traffic to alternative paths. Hence, the multi-path BGP also has a potential advantage against denial-of-service attack.

5.2.2 Security Extension

BGP is vulnerable to malicious actions such as an advertisement of a fake path, because BGP speakers fully depend on the routing information send from neighboring speakers. Hence, authentication mechanisms are required to guarantee the validity of route advertisements. Origin authentication considers whether the originating AS controls the claimed IP address ranges. Path authentication confirms that all the ASes are authorized to advertise the routes to destination IP address blocks. S-BGP and BGPSEC were proposed as such protocols. Address attestations are for origin authentica-

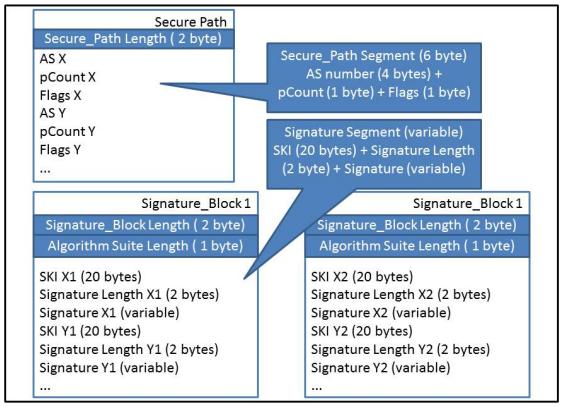


Figure 5.4: BGPSEC_path Attribute

tion, and route attestations are for path authentication. To support signing and verification processes, S-BGP and BGPSEC require PKI. In particular, ASes have their own X.509 as certificates of public keys. A route attestation is signed by a BGP speaker to authenticate the existence and position of an AS number in an AS path. Such an attestation is nested, and each BGP speaker signs the AS path in sequence. At first, the speaker signs the origin BGP speaker signs the AS number of the origin autonomous system, the prefix, and the intended receiver. The next signer is the recipient of this route attestation, and signs the concatenation of the new AS path, the prefix, and intended recipient. The process goes on until the entire AS path is signed.

We give the detail of a packet of BGPSEC in Fig. 5.2.2. According to [58], the construction is almost the same as that of BGP-4, which is the conventional BGP, except for AS_path attribute information are replaced to BGPSEC_path attribute in Fig. 5.2.2.

5.3 Aggregate Signatures

we describe several related works in terms of ID-based aggregate signatures and structured signatures.

5.3.1 Exiting Schemes

Although many aggregate signature schemes have been proposed [1, 6, 9, 17, 35, 55, 54, 64, 66, 71, 85], there are few ID-based aggregate signature schemes [6, 14, 15, 31, 36, 94]. The famous schemes are the general scheme by Bagherzandi and Jarecki [6], the synchronized scheme by Gentry and Ramzan [36] and the sequential scheme by Boldyreva et al. [14, 15], where the scheme in [14] was broken by Hwang et al. [45] but the bug was fixed in [15]. Unfortunately, all of these schemes except for the scheme by Wang et al. [94] cannot deal with a complicated structure such as that in structured signature schemes. Only the scheme by Wang et al. can deal with the structures, but, this scheme is remarkably inefficient in the sense that both the signature size and the number of computations of the bilinear maps increase linearly. Hence, we judge that there is, in essence, no ID-based structured aggregate signature scheme. We note that an open problem of ID-based aggregate signatures is to construct a scheme without random oracles.

Whereas researches for ID-based aggregate structured signatures are few, several schemes have been reported in the framework of the conventional public key cryptography [19, 30, 61, 62, 90, 100]. The oldest one was an RSA-based structured signature scheme by Doi et al. [30], and then Burmester et al. [19] proposed the first DLP-based scheme. These schemes hierarchically classify their handling structures, i.e., a serial structure as a relation among vertical relationships and a parallel structure as a relation among horizontal relationships, and these schemes can deal with the same structures as that of our scheme. In this approach, Mitomi and Miyaji [68] defined the order flexibility as a desirable property. In the schemes with no order flexibility, the number of signers are drastically restricted. To the author's knowledge, the schemes meeting this capability have been proposed only in [90, 100]. Among them, the scheme with the fixed size signature is only the one by Yamamoto and Ogata [100]. Nevertheless, the scheme is inefficient because the number of computations of the bilinear maps increases linearly.

5.3.2 Application to BGP Security

Since aggregate signatures allow each signer to sign an individual document and to compress these signatures, many researchers have focused on BGP as its key application. For instance, by utilizing aggregation signatures in BGP, BGPSEC path attribute can be written as Fig. 5.5. The size of the signatures which is the main problem in the conventional schemes becomes

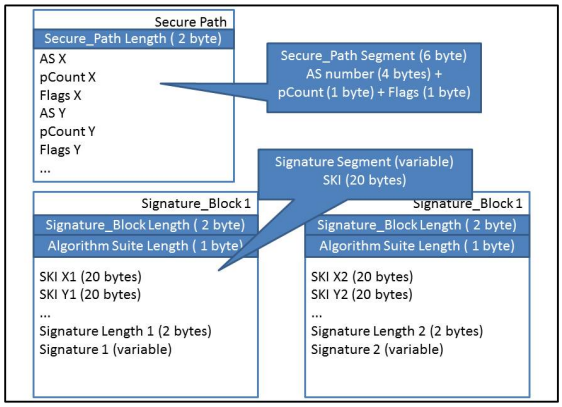


Figure 5.5: BGPSEC path Attribute with Aggregate Signatures

fixed, and so a large number of paths can be constructed.

5.4 BGP-Aiding Aggregate Signatures

In this section, we describe the existing gap between the recent routing security and aggregate signatures and a notion of BGP-aiding aggregate signatures. Then, we define a syntax of a BGP-aiding aggregate signature scheme and its security. In this chapter, we utilize the following notations. We denote by $\mathcal{M}$ an message space and by $\mathcal{ID}$ an ID space. We also denote by $ID_i \in \{0,1\}^*$ the ith signer, by $m_i \in \{0,1\}^*$ message to be signed by ID_i, by σ_i a signature generated by ID_i, by mpk a master public key, by msk a master secret key, and by sk_i a secret key of ID_i.

5.4.1 Technical Gaps between Routing Security and Recent Aggregate Signatures

Recent researches for aggregate signatures aim to construct sequential aggregate signatures [15, 18, 35, 37, 54, 55]. Unlike other types of aggregate signatures, the sequential aggregate signatures allow signers to avoid to broadcast signatures, and the traffic of the network can be decreased. Even with such a positive property, we judge the sequential aggregate signatures are unsuitable in BGP under the multi-path setting. In particular, a sequential aggregate signature scheme utilizes aggregate-signing where each signer signs and aggregates at the same time instead of an ordinary signing

Table 5.1: Evaluation of Proposed Scheme

We compare the performance of the proposed scheme with the existing ID-based aggregate signature schemes with respect the computational cost for ith signer, the computational cost for a verifier with n signers and the signature size. We denote by $\mathcal{P}$ the computational cost of a bilinear map, by $\mathcal{E}$ the computational cost of an exponentiation computation, by $\mathcal{H}$ the computational cost of a map-to-point, by $L(p)$ the binary length of a prime number p, by $L(N)$ the binary length of a composite number N, and by k as a security parameter. Typical values for these parameters are $\mathcal{L}(p) = 176$ on a symmetric pairing for 80-bit security, $\ell = 176$, $n = 20$, and, with Type A curve in PBC library [65] according to [47], the cost per one $\mathcal{P}$ is 2.2078 msec, the cost per one $\mathcal{E}$ is 2.5591 msec and the cost per one $\mathcal{H}$ is 5.8960 msec. For Multihoming, "Applicable" means that the scheme is applicable to multihoming, but "Not" means that the scheme is not applicable.

Relate Work	Signature Size	Number of Rounds	Multihoming	Cryptosystem
BG10 [6]	$2L(N) + 2k + \log n$	2	Applicable	IBC
BGLS03 [17]	$L(p)$	1	Applicable	PKI
BGOY10 [15]	$3L(p)$	-	Not	IBC
DZXH09 [31]	$(n+1)L(N)$	-	Not	IBC
GR06 [36]	$2L(p) + k$	1	Applicable	IBC
GLOW12 [37]	$5\mathcal{L}(N)$	-	Not	IBC
WOMOD07 [94]	$2nL(p)$	-	Applicable	IBC
Modified GR06	$2L(p) + k$	1	Applicable	IBC

algorithm. In other words, the sequential aggregate signature scheme requires each signer to provide a secret key. In this case, while an individual signature is generated for each path, these signatures cannot be aggregated. Thus, we consider that the sequential aggregate signature scheme has a gap from a realistic network environment. Meanwhile, in order to remove the restriction in the number of ASes from BGP, we have to consider subject key identifiers which become a new bottleneck.

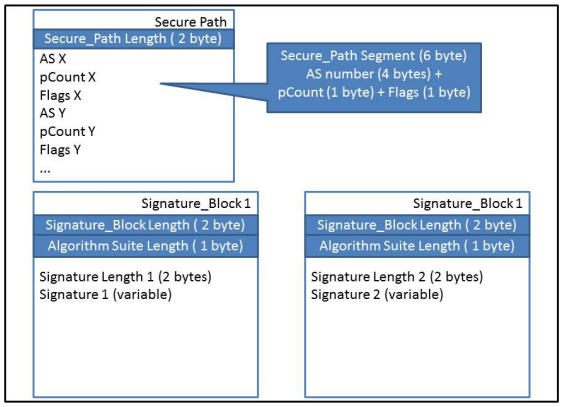


Figure 5.6: BGPSEC path Attribute with ID-based Aggregate Signatures

5.4.2 Requirements of BGP-Aiding Aggregate Signatures

As described in the previous section, the main discussion of aggregate signatures for BGP is *how to aggregate* signatures for different paths. In this section, we discuss capability required in BGP of aggregate signatures, and review the existing schemes.

ID-based Signature Scheme The packet size of BGP is restricted in 4,096 bytes, and the total size of data including signatures and subject key identifiers must be smaller than this value. In the conventional public key cryptosystem, a public key is given as a random string and a subject key identifier is necessary in order to distinguish the public key of each user. In general, the size of the subject key identifiers is twenty bytes and is ignored for constructing a cryptosystem. However, these data become a problem for a small packet space such as BGP, and so these should be removed. As known cryptographic primitives, we can utilize ID-based cryptography [86]. For instance, by utilizing an ID-based signature scheme, each AS can utilize its AS number as a public key. This approach allows ASes to remove the subject key identifiers from the packets, and hence the packets can be written in Fig. 5.4.2.

We also discuss the advantage of ID-based cryptography from the aspect of an operation side. ID-based cryptography is sometimes misunderstood as a method to remove certificates since any string can be utilized as a public key. The certificates are used in order to specify a utilized cryptographic

algorithm and its parameter, and so are necessary even if an ID-based cryptosystem is utilized [49]. Meanwhile, since a key generation center cannot choose a string corresponding to a public key, constructing trust in a relation between each user and its corresponding ID information is a crucial task [77]. Nevertheless, we note that the certificates become unnecessary in BGP security by utilizing the ID-based cryptosystem, because in this framework ASes can utilize a standardized data format specified for BGP packets. In addition, AS numbers are decided by IANA in advance, and trust in the relation between users and ID can be established via the trust in IANA.

General Aggregation General aggregation allows any third party to aggregate generated signatures, and ordinary schemes such as the first aggregate signature scheme [17] adopts the method. Intuitively, the method requires an interactive process of signers, and is considered as an inefficient approach. Hence, in recent researches an alternative method, sequential aggregation, has been studied [15, 18, 35, 54, 55, 71]. Among these two methods, we judge that general aggregation is suitable for the multi-path: in particular, each AS can combine signatures given from each path and then can sign path information of the combined networks.

5.4.3 Definition of Signer Structures

In this subsection, we define structures of signers. The definition is almost the same as that in Chapter 4, and recall the definition. We define two structures as the relation for a group of signers, serial structure and parallel structure, and consider a series-parallel graph described in Chapter 2 as a signer structure. Here, an edge of a series-parallel graph corresponds to a signer, and a unique edge for the graph corresponds to a unique index representing the position of each signer in any structure, i.e., an index i in ψ_n corresponds to ID_i which is the ith signer in the signer structure. Here, we also give each edge a space storing a message to be signed together with an ID of each signer [1]. Hereafter, we denote by ψ_n a signer structure consisting of n signers, by $\mathcal{T}(i)$ a set of signers connecting to the initial vertex I_i of ith signer in a way such that $\mathcal{T}(i) = \{x | I_i = T_x \wedge 1 \leq x < i \wedge G_x(I_x, T_x) \subset \psi_n\}$, by $\mathcal{I}(i)$ a set of signers connecting to the terminal vertex T_i of ith signer in a way such that $\mathcal{I}(i) = \{x | T_i = I_x \wedge i < x \leq n \ \wedge G_x(I_x, T_x) \subset \psi_n\}$, by $\{a_j\}_{j \in \mathcal{T}(i)}$, for all a, all a_j for $j \in \mathcal{T}(i)$. We also define an operation $\dot{\subset}$ where $i \dot{\subset} \psi_n$ means extractions of indexes from ψ_n.

[1] This graph can be implemented via both a two-dimensional array representing relations among the graph of the graph and an array to store the messages.

5.4.4 The Syntax

A BGP-aiding aggregate signature scheme consists of the following four algorithms:

Setup Given a security parameter 1^k, output a master secret key msk and its corresponding master public key mpk.

KeyGeneration Given msk, mpk and a signer's ID information ID_i, output a secret key sk_i.

Signing Given mpk, sk_i, a message $m_i \in \mathcal{M}$, an ID $ID_i \in \mathcal{ID}$, for $k \in \mathcal{T}(i)$ the previous signers' structures ψ_k consisting of $\#\psi_k$ signers with $\#\psi_k$ messages and their signatures $\sigma_k = (S_k, R_k)$, and possibly some state information s as input, compose $\psi_{\mathcal{T}(i)} := \cup_{k \in \mathcal{T}(i)} \psi_k$ and then compose $\psi_i := \psi_{\mathcal{T}(i)} \cap \phi_i$ where ϕ_i contains m_i and ID_i. Generate an aggregate signature σ_i in the composed structure ψ_i with the signatures $\{\sigma_j\}_{j \in \mathcal{T}(i)}$, and output (ψ_i, s, σ_i). If any input is an error, then output an error symbol $\perp$.

Verification Given $(mpk, \{ID_i\}_{i \dot{\subset} \psi_n}, \psi_n, s, \sigma_n)$ as input, output *accept* or *reject*.

5.4.5 Definition of the Security

The following model allows an adversary $\mathcal{A}$ to execute *adaptively-chosen-message attack and adaptively-chosen-identity attack* [23]. This model is a natural extension of the existing ID-based aggregate signature schemes [14, 36]. In this model, players are a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ with a security parameter 1^k as input. We denote by $x^{(h)}$ h-th query for all x in the following game:

Setup $\mathcal{C}$ runs **Setup** to obtain msk and mpk, and gives mpk to $\mathcal{A}$.

Key Generation Given any information $ID_i^{(h)} \in \mathcal{ID}$ chosen by $\mathcal{A}$, $\mathcal{C}$ generates
$sk_i \leftarrow$ **KeyGeneration**(msk, mpk, ID_i) and returns $sk_i^{(h)}$.

Signing Query $\mathcal{A}$ chooses a state information s, i signers $\{ID_j^{(h)}\}_{j \in [1,i]}$ and i messages $\{m_j^{(h)}\}_{j \in [1,i]}$, and decides a signer structure $\psi_i^{(h)}$. Then, $\mathcal{A}$ generates $(\{ID_j^{(h)}\}_{j \dot{\subset} \psi_k}, \psi_k^{(h)}, \sigma_k, s)$ for $k \in \mathcal{T}(i)$, and sends a query

$\left(m_i^{(h)}, \left\{(\{ID_j^{(h)}\}_{j \dot{\subset} \psi_k}, \psi_k^{(h)}, \sigma_k)\right\}_{k \in \mathcal{T}(i)}, s\right)$ to $\mathcal{C}$. Then, generate $\sigma_i \leftarrow$ **Signing** $\left(mpk, sk_i, m_i^{(h)}, ID_i^{(h)}, \left\{(\psi_k^{(h)}, \sigma_k)\right\}_{k \in \mathcal{T}(i)}, s\right)$. $\mathcal{C}$ returns σ_i.

Output $\mathcal{A}$ outputs a forgery σ^* in ψ_n^* consisting of $\{ID_i^*\}_{i=1}^n$ and $\{m_i^*\}_{i=1}^n$ with a state information s^*. $\mathcal{C}$ checks that the following conditions hold; **Verification**$((mpk, \{ID_i^*\}_{i \dot{\subset} \psi_n} \psi_n^*, s^*, \sigma_n^*))$ outputs *accept*; For some i, $1 \leq i \leq n$, $\mathcal{A}$ did not query $ID_{i^*}^*$ to **KeyGeneration**; Each ID_i^* does not appear more than once in ψ_n^*; For some i, $1 \leq i \leq n$, $\mathcal{A}$ did not query $(m_{i^*}^*, \{\psi_k^*\}_{k \in \mathcal{T}(i^*)})$ and s to **Signing**. $\mathcal{A}$ wins if all of these conditions hold. Here, we do not consider that $\mathcal{A}$ wins if s^* has been queried in **Signing**, even if $(m_{i^*}^*, \{\psi_k^*\}_{k \in \mathcal{T}(i^*)})$ has never been queried.

Definition 19. An adversary $\mathcal{A}$ breaks a BGP-aiding aggregate signature scheme with $(t, q_s, q_k, q_h, n, \epsilon)$ if $\mathcal{A}$ who does not know msk can win the above game with a success probability greater than ϵ within an execution time t. Here, $\mathcal{A}$ can query to **Signing** with $\mathcal{C}$ at most q_s times, **KeyGeneration**$(msk, \cdot)$ at most q_k times and q_h random oracle queries, and n is the number of signers included in the forgery. We say that an aggregate signature scheme is $(t, q_s, q_k, q_h, n, \epsilon)$-secure if there is no adversary who breaks the scheme with $(t, q_s, q_k, q_h, n, \epsilon)$.

5.5 Instantiation of the Signature Scheme

We focus on the GR06 scheme [36]. This scheme is an ID-based aggregate signature scheme and closes to the requirements of the BGP-aiding aggregate signature scheme. In this section, we propose the modified-GR06 scheme as an instantiation of BGP-aggregate signature scheme, and prove the security.

5.5.1 Construction of the Modified-GR06 Scheme

Our scheme is a synchronized scheme, and uses a state information s as one-time information similarly as the schemes in [1, 36].

Setup Given a security parameter 1^k, generate $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ as a pairing parameters, and choose a generator $g \in \mathbb{G}$ and a random number $\alpha \in \mathbb{Z}_p$. Then set $A = g^\alpha$, and choose hash functions $H_1 : \{0,1\}^* \times$

$\{0,1\} \to \mathbb{G}$, $H_2 : \{0,1\}^* \to \mathbb{G}$ and $H_3 : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_p^*$. Finally, output $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, A, H_1, H_2, H_3)$ as mpk and α as msk.

Key Generation Given signer's ID_i, compute values $g_{i,j}^{\alpha}$ where $g_{i,j} = H_1(ID_i, j)$ for $j = \{0,1\}$, and return these values as a secret key sk_i of ID_i.

Signing Given $(ID_i, m_i, \{\sigma_j\}_{j\in\mathcal{T}(i)}, \{\psi_j\}_{j\in\mathcal{T}(i)}, s)$, check the validity of the given signatures by using **Verification** with $n = j$ for $j \in \mathcal{T}(i)$. If any output is invalid, output $\perp$. Otherwise, compose $\psi_i = \left(\cup_{j\in\mathcal{T}(i)}\psi_j\right) \cap \phi_i$ where ϕ_i stores ID_i and m_i, and compute $g_s = H_2(s)$ and $c_i = H_3(ID_i, \psi_i, s)$. Then, generate a secret random number $r_i \leftarrow \mathbb{Z}_p$ and compute as follows:

$$S_i \leftarrow g_s^{r_i} g_{i,0}^{\alpha}(g_{i,1}^{\alpha})^{c_i} \prod_{j\in\mathcal{T}(i)} S_j, \quad R_i \leftarrow g^{r_i} \prod_{j\in\mathcal{T}(i)} R_j.$$

If ID_i is the first signer, set $\psi_1 = \phi_1$ and $\{\sigma_j\}_{j\in\mathcal{T}(1)} = 1$, and choose s that it has never used before. The signature is $\sigma_i = (S_i, R_i, s)$. Then output (ψ_i, s, σ_i). If there are k signers $\{ID_k\}_{k\in\mathcal{T}(n)}$ as the last signers of the whole signing group, then the whole structure is $\psi_n = \cup_{k\in\mathcal{T}(n)}\psi_k$ and the whole signature is $(S_n = \prod_{k\in\mathcal{T}(n)} S_k, R_n = \prod_{k\in\mathcal{T}(n)} R_k, s)$.

Verification Given $(mpk, \{ID_j\}_{j\dot{\subset}\psi_n}, \psi_n, s, \sigma_n)$, compute as follows:

$$\boldsymbol{e}(S, g) \stackrel{?}{=} \boldsymbol{e}(R, g_s)\boldsymbol{e}\left(A, \prod_{i=1}^{n} \left(g_{i,0}(g_{i,1})^{c_i}\right)^{\omega_i(\psi_n)}\right),$$

where $g_s = H_2(s)$, $g_{i,j} = H_1(ID_i, j)$ for $j \in \{0,1\}$ and $c_i = H_3(ID_i, \psi_i, s)$. If the above equation holds, then output *accept*. Otherwise, output *reject*.

5.5.2 Security Proof of the Modified-GR06 Scheme

In this subsection, we prove the security of proposed scheme. In this proof, we reduce the security of the hardness of the CDH problem in $\mathbb{G}$. The main proof is similar as that in [36].

Theorem 20. Suppose that the (t', ϵ')-CDH assumption holds in $\mathbb{G}$ and hash functions are modeled as random oracles. Then the proposed scheme is $(t, q_s, q_k, q_{h_1}, q_{h_2}, q_{h_3}, n, \epsilon)$-secure, where

$$t = t' - (5q_s + 2q_k + 2q_{h_1} + q_{h_2})\, t_e + \mathcal{O}(n), \;\; \epsilon = \epsilon' \frac{e^3\left(q_k + q_{h_1}(q_s + q_{h_3}) + 3\right)^3}{27},$$

e is the base of natural logarithm and t_e is the computational time for one exponentiation computation.

Sketch. We assume that an adversary $\mathcal{A}$ who breaks the proposed scheme with $(t, q_s, q_k, q_{h_1}, q_{h_2}, q_{h_3}, n, \epsilon)$ exists, and then construct an algorithms $\mathcal{B}$ to solve the CDH problem in $\mathbb{G}$ by utilizing $\mathcal{A}$ as a subroutine. In this proof, we utilize the Coron technique [27]: in simulations of the random oracles, $\mathcal{B}$ uses random coins with some probability δ to set 1 on these coins, and δ is optimized later in order to complete the proof. $\mathcal{B}$ also has lists of the random oracles, H_1-list $[\cdot, \cdot, \cdot, \cdot, \cdot, \cdot]$, H_2-list $[\cdot, \cdot, \cdot, \cdot]$ and H_3-list $[\cdot, \cdot, \cdot, \cdot, \cdot, \cdot]$. Given a challenge $(g, g^a, g^b, p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ of the CDH problem, the CDH challenge for short, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup Set $A = g^a$ and send $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, A, H_1, H_2, H_3)$ as mpk to $\mathcal{A}$.

H_1**-Hash Query** Given (ID_i, j) chosen by $\mathcal{A}$ for $j = \{0, 1\}$, execute the following steps. Here, if ID_i was already queried as H_1-query, retrieve $g_{i,j}$ from H_1-list. Generate a random coin $H_1\text{-coin}_i \leftarrow \{0, 1\}$ with the probability δ and random numbers $\gamma_{i,j,1}, \gamma_{i,j,2} \leftarrow \mathbb{Z}_p$. If $H_1\text{-coin}_i = 0$, then compute $g_{i,j} = g^{\gamma_{i,j,1}}$; otherwise, compute $g_{i,j} = g^{\gamma_{i,j,1}}(g^b)^{\gamma_{i,j,1}}$. Then, record $(H_1\text{-coin}_i, ID_i, j, \gamma_{i,j,1}, \gamma_{i,j,2}, g_{i,j})$, and return $g_{i,j}$ to $\mathcal{A}$.

H_2**-Hash Query** Given s chosen by $\mathcal{A}$, execute the following steps. If s was already queried as H_2-query, retrieve g_s from H_2-list. Else, generate a random coin $H_2\text{-coin}_k \leftarrow \{0, 1\}$ with the probability δ and a random number $\beta \leftarrow \mathbb{Z}_p$. If $H_2\text{-coin}_k = 0$ then set $g_s = (g^b)^\beta$; otherwise, set $g_s = g^\beta$. Record $(H_2\text{-coin}_k, s, \beta, g_s)$ and return g_s to $\mathcal{A}$.

H_3**-Hash Query** Given (ID_i, ψ_i, s) chosen by $\mathcal{A}$, execute the following steps. If (ID_i, ψ_i, s) was already queried as H_3-query, retrieve c_i from H_3-list. Otherwise, check that $H_1\text{-coin}_i$ and $H_2\text{-coin}_j$ for the given query at first. If H_1**-Hash Query** and H_2**-Hash Query** have been never executed for the given query, execute these oracle simulations before. Generate a random coin $H_3\text{-coin}_l \leftarrow \{0, 1\}$ with the probability δ, and execute as follows:

1. If $H_1\text{-coin}_i = H_2\text{-coin}_j = 1$ and $H_3\text{-coin}_l = 0$, check that H_3-list contains (ID'_i, ψ'_i, s') such that $(ID_i, \psi_i, s) \neq (ID'_i, \psi'_i, s')$ holds with $ID_i = ID'_i$. If so, abort the process. Otherwise, set $c_i = -\frac{\gamma_{i,0,2}}{\gamma_{i,1,2}}$.
2. If $H_1\text{-coin}_i = 0$ or $H_2\text{-coin}_j = 0$ or $H_3\text{-coin}_l = 1$, generate a random number $d_i \leftarrow \mathbb{Z}_p$ and set $c_i = d_i$

Record $(H_3\text{-coin}_l, ID_i, \psi_i, s, d_i, c_i)$ in H_3-list and return c_i to $\mathcal{A}$.

Key Generation Given ID_i by $\mathcal{A}$, check $H_1\text{-coin}_i$ for ID_i in H_1-list. If ID_i has never been queried, execute H_1**-Hash Query** with ID_i. If $H_1\text{-coin}_i = 0$ then set $g_{i,0}^a = A^{\gamma_{i,0,1}}$ and $g_{i,1} = A^{\gamma_{i,1,1}}$, and return $g_{i,j}^a$ where $j = \{0, 1\}$. Otherwise, abort the process.

Signing Given $(ID_i, m_i, \{\sigma_k\}_{k\in\mathcal{T}(i)}, \{\psi_i\}_{k\in\mathcal{T}(i)}, s)$ by $\mathcal{A}$, execute the following steps. Here, if the verification algorithm outputs $reject$ for any signature in the given query, return $\perp$. Otherwise, execute the following steps:

- If $H_1\text{-coin}_i = 0$ for ID_i in H_1-list, then generate a secret random number $r_i \leftarrow \mathbb{Z}_p$, and compute $R_i \leftarrow g^{r_i} \prod_{k\in\mathcal{T}(i)} R_k$, $c_i = H_3(ID_i, \psi_i, s)$ and $S_i \leftarrow A^{\gamma_{i,0,1}}(A^{\gamma_{i,1,1}})^{c_i} g_s^{r_i} \prod_{k\in\mathcal{T}(i)} S_k$. Here, $A^{\gamma_{i,j,1}} = g_{i,j}^a$ holds for $j = \{0, 1\}$, and hence these values are accepted as a valid signature. Return (S_i, R_i, s).
- If $H_1\text{-coin}_i = 1$ and $H_2\text{-coin}_k = 0$, then generate a secret random number $r_i \leftarrow \mathbb{Z}_p$, and compute $R_i \leftarrow g^{r_i}(g^a)^{-\frac{(\gamma_{i,0,2}+\gamma_{i,1,2}c_i)}{\beta}} \prod_{k\in\mathcal{T}(i)} R_k$, $c_i = H_3(ID_j, \psi_i, s)$ and $S_i \leftarrow A^{\gamma_{i,0,1}}(A^{\gamma_{i,1,1}})^{c_i} g_s^{r_i} \prod_{k\in\mathcal{T}(i)} S_k$. Here, these values can be written as follows:

$$\begin{aligned} S_i &= (g^{\gamma_{i,0,1}})^a \left((g^{\gamma_{i,1,1}})^a\right)^{c_i} \left((g^b)^{\gamma_{i,0,2}-\gamma_{i,0,2}}\right)^a \left((g^b)^{\gamma_{i,1,2}-\gamma_{i,1,2}}\right)^{ac_i} \\ &\quad \times g_s^{r_i} \prod_{k\in\mathcal{T}(i)} S_k \\ &= \left(g^{\gamma_{i,0,1}}(g^b)^{\gamma_{i,0,2}}\right)^a \left(g^{\gamma_{i,1,1}}(g^b)^{\gamma_{i,1,2}}\right)^{ac_i} \left((g^b)^\beta\right)^{r-\frac{a(\gamma_{i,0,2}+\gamma_{i,1,2}c_i)}{\beta}} \\ &\quad \times \prod_{k\in\mathcal{T}(i)} S_k \\ &= g_{i,0}^a (g_{i,1}^a)^{c_i} g_s^{r-\frac{a(\gamma_{i,0,2}+\gamma_{i,1,2}c_i)}{\beta}} \prod_{k\in\mathcal{T}(i)} S_k. \end{aligned}$$

  Hence, these values are accepted as a valid signature. Return (S_i, R_i, s).
- If $H_1\text{-coin}_i = H_2\text{-coin}_k = 1$ and $H_3\text{-coin}_l = 0$, then generate a secret random number $r_i \leftarrow \mathbb{Z}_p$, and compute $R_i \leftarrow g^{r_i} \prod_{k\in\mathcal{T}(i)} R_k$, $c_i = H_3(ID_i, \psi_i, s)$ and $S_i \leftarrow A^{\gamma_{i,0,1}}(A^{\gamma_{i,1,1}})^{c_i} g_s^{r_i} \prod_{k\in\mathcal{T}(i)} S_k$. Here,

these values can be written as follows:

$$\begin{aligned} S_i &= (g^{\gamma_{i,0,1}})^a \left((g^{\gamma_{i,1,1}})^a\right)^{-\frac{\gamma_{i,0,2}}{\gamma_{i,1,2}}} \left((g^b)^{\gamma_{i,0,2}}\right)^a \left((g^b)^{-\gamma_{i,1,2}\frac{\gamma_{i,0,2}}{\gamma_{i,1,2}}}\right)^a \\ &\quad \times g_s^{r_i} \prod_{k\in\mathcal{T}(i)} S_k \\ &= \left(g^{\gamma_{i,0,1}}(g^b)^{\gamma_{i,0,2}}\right)^a \left(\left(g^{\gamma_{i,1,1}}(g^b)^{\gamma_{i,1,2}}\right)^a\right)^{-\frac{\gamma_{i,0,2}}{\gamma_{i,1,2}}} g_s^r \prod_{k\in\mathcal{T}(i)} S_k \\ &= g_{i,0}^a (g_{i,1}^a)^{c_i} g_s^r \prod_{k\in\mathcal{T}(i)} S_k. \end{aligned}$$

Hence, these values are accepted as a valid signature. Return (S_i, R_i, s).

- If $H_1\text{-coin}_i = H_2\text{-coin}_k = H_3\text{-coin}_l = 1$, then abort the process.

Output Given a forgery $(\{ID_i\}_{i\dot{\subset}\psi_n^*}, \psi_n^*, s^*, \sigma_n^*)$ by $\mathcal{A}$, check that for the given forgery any indexes (i,k,l) exist such that $H_1\text{-coin}_i = H_2\text{-coin}_k = H_3\text{-coin}_l = 1$ holds. If not, abort the process. Otherwise, compute as follows, where Δ is a set of indexes such that $H_1\text{-coin}_i = 1$:

$$\begin{aligned} &\left(\frac{S^*}{(R^*)^\beta A^{\sum_{i=1\wedge i\notin\Delta}^n((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)} A^{\sum_{i\in\Delta}((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)}}\right)^X \\ = &\left(\frac{g_s^r \prod_{i=1}^n \left(g_{i,0}^a (g_{i,1}^a)^{c_i}\right)^{\omega_i(\psi_n)}}{(R^*)^\beta A^{\sum_{i=1\wedge i\notin\Delta}^n((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)} A^{\sum_{i\in\Delta}((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)}}\right)^X \\ = &\left(\frac{(g^\beta)^r \prod_{i=1}^n \left(g_{i,0}^a (g_{i,1}^a)^{c_i}\right)^{\omega_i(\psi_n)}}{(g^r)^\beta A^{\sum_{i=1\wedge i\notin\Delta}^n((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)} A^{\sum_{i\in\Delta}((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)}}\right)^X \\ = &\left(\frac{\prod_{i\in\Delta} \left(g_{i,0}^a (g_{i,1}^a)^{c_i}\right)^{\omega_i(\psi_n)}}{A^{\sum_{i\in\Delta}((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)}}\right)^X \\ = &\left(\frac{\prod_{i\in\Delta} \left(\left(g^{\gamma_{i,0,1}}(g^b)^{\gamma_{i,0,2}}\right)^a \left(\left(g^{\gamma_{i,1,1}}(g^b)^{\gamma_{i,1,2}}\right)^{ac_i}\right)\right)^{\omega_i(\psi_n)}}{A^{\sum_{i\in\Delta}((\gamma_{i,0,1}+c_i\gamma_{i,1,1})\omega_i(\psi_n^*)}}\right)^X \\ = &\ g^{ab}, \end{aligned}$$

where $X = \frac{1}{A^{\sum_{i\in\Delta}\left((\gamma_{i,0,2}+c_i\gamma_{i,1,2})\omega_i(\psi_n^*)\right)}}$.

$\mathcal{B}$'s success probability ϵ' can be obtained as follows. Let $abort_{h_3}$ be the event that $\mathcal{B}$ aborts for H_3-**Hash Query**, $abort_k$ be the event that $\mathcal{B}$ aborts for **Key Generation**, $abort_s$ be the event that $\mathcal{B}$ aborts for **Signing**, and $abort_o$ be the event that $\mathcal{B}$ aborts for **Output**. Each probability can be obtained as follows:

$$\begin{aligned}
\Pr[\overline{abort_{h_3}}] &= ((1-\delta)^{q_{h_1}} + (1-\delta)^{q_{h_2}})^{q_{h_3}} \geq (1-\delta)^{q_{h_1} q_{h_3}}, \\
\Pr[\overline{abort_k}] &= (1-\delta)^{q_k}, \\
\Pr[\overline{abort_s}] &= ((1-\delta)^{q_{h_1}} + (1-\delta)^{q_{h_2}} + (1-\delta)^{q_{h_3}})^{q_s} \geq (1-\delta)^{q_{h_1} q_s}, \\
\Pr[\overline{abort_o}] &= \delta^3.
\end{aligned}$$

These are independent events, and $\mathcal{B}$ can always solve the CDH problem in $\mathbb{G}$ if any event does not occur. Therefore, the following equation holds:

$$\begin{aligned}
\epsilon' &= \epsilon \Pr[\overline{abort_{h_3}}] \Pr[\overline{abort_k}] \Pr[\overline{abort_s}] \Pr[\overline{abort_o}] \\
&\geq \epsilon (1-\delta)^{q_{h_1} q_{h_3}} (1-\delta)^{q_k} (1-\delta)^{q_{h_1} q_s} \delta^3
\end{aligned}$$

Finally, we optimize δ. We define a function $f(\delta) = (1-\delta)^{q_{h_1} q_{h_3}} (1-\delta)^{q_k} (1-\delta)^{q_{h_1} q_s} \delta^3$. By computing its derived functions, the function has an extremum $\delta_{opt} = \frac{3}{q_{h_1}(q_s+q_{h_3})+q_k+3}$. Then, the following equation holds from the definition of δ.

$$\begin{aligned}
\epsilon' &\geq \epsilon \cdot f(\delta_{opt}) \\
&= \epsilon \cdot \left(\frac{q_{h_1}(q_s+q_{h_3})+q_k}{q_{h_1}(q_s+q_{h_3})+q_k+3} \right)^{q_{h_1}(q_s+q_{h_3})+q_k} \left(\frac{3}{q_{h_1}(q_s+q_{h_3})+q_k+3} \right)^3
\end{aligned}$$

From the definition of the base of natural logarithm, i.e., $\lim_{x\to\infty} \left(\frac{x}{x+1}\right)^x = \frac{1}{e}$, we can obtain the probability as follows:

$$\epsilon' \geq \epsilon \cdot \frac{1}{e^3} \cdot \frac{27}{(q_{h_1}(q_s+q_{h_3})+q_k+3)^3}.$$

The execution time of $\mathcal{B}$ is that of $\mathcal{A}$ plus two exponentiations for H_1-**Hash Query**, one exponentiation for H_2-**Hash Query**, two exponentiations for **Key Generation**, at most five exponentiations for **Signing** and four exponentiations for **Output**. Thus, the following equation holds:

$$t \geq t + (2q_{h_1} + q_{h_2} + 2q_k + 5q_s + 4)t_e,$$

where t_e is the computational time per exponentiation. □

Restriction on the Number of Signers

There is a possibility of the attack-0 described in Chapter 3 by utilizing the property of the graph theorem against the proposed scheme. In particular, colluding adversaries can cancel out a signature of a target signer from an aggregate signature by generating a graph ψ_n such that $\omega_i(\psi_n) = 0 \bmod p$ holds. However, this probability can be easily estimated by the graph theorem for all graph. As described in Chapter 2, the weight of the graph consisting of n edges is bounded by $3^{n/3}$ [90], and hence $n = \frac{\log p}{\log 3}3$ can be obtained. Thus, the value is about 300 for 80-bit security, and the value for 128-bit security is about 969. Namely, the attack described above can be prevented as long as the number of signers are less than these values. One might think that this construction is insufficient since restriction in the number of signers is necessary. However, we describe the restriction does not become a problem in a practical scenario.

5.6 Discussion

Table 5.2: AS Numbers for each RIR

(APNIC) [3]	ARIN	LACNIC	RIPE NCC	AfriNIC	IANA	Undecided
7830	25428	3839	25112	1277	1042	1008

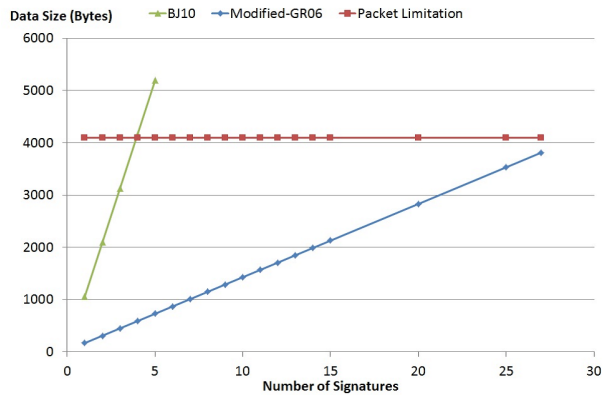


Figure 5.7: Evaluation of ID-based Aggregate Signatures

For 16-bit AS number, there are 65,535 ASes. All of these numbers are not uniformly managed but each regional Internet registry (RIR) redistributes its AS numbers to subnetworks. In an actual routing security, RIR redistributed AS numbers and corresponding certificates via resource public key infrastructure (RPKI) [57], and each AS executes the signing process by the given resource certificate [58]. Namely, there is no problem in a practical scenario as long as the total size of signatures obtained from the maximum number of signers for each RIR is smaller than 4,096 bytes. The number of ASes managed by each RIR in May 2013 is shown in Table. 5.2 [46].

The maximum number of the signers is given by the weight of the graph as described in the previous section. For 128-bit security, the maximum number of the signers given by the modified-GR06 scheme is 969, and hence 27 signatures are generated in ARIN whose number of ASes is 25428. In Fig. 5.6, we give the communication cost. From this figure, we consider that available BGP-aiding aggregate signature scheme is only the modified-GR06 scheme.

Chapter 6

Certificateless Aggregate Signatures

6.1 Introduction

6.1.1 Motivation of CAS

Identity-based aggregate signatures provide many important applications such as BGP security while these signatures have an inherent problem, called *key escrow problem*, in which a key generation center knows secret keys of all users in the system. This problem occurs because secret keys of all the users are computed from KGC's master secret key and users' ID. This implies that the KGC must be trusted in ID-based cryptosystem. In general, an establishment of trust for an authority is a difficult task [78]. In fact, ENISA pointed out that the most important threat in a cloud environment is an existence of a malicious provider [22]. For instance, malicious KGC's who does not honestly run the algorithm exist[56]. Hence, developments of identity-based cryptosystem have been disturbed [77].

In order to overcome this problem, Al-Riyami and Paterson proposed the *certificateless cryptosystem* [2] which is a hybrid cryptosystem of PKI and ID-based cryptosystem. In the certificateless cryptosystem, keys of each user consists of a pair of secret key and public key depending upon both PKI and ID-based cryptosystem. In particular, after given a secret value in ID-based cryptosystem called partial private key, each user generates a secret value which is a random number in PKI. Then the user sets a full secret key from the secret value and the partial private key, and then sets a value computed from the secret value in PKI and its ID as a corresponding public

key. [1] A sender/verifier uses the public key for the encryption/verification of data, and a receiver/signer uses the full secret key for the decryption/signing of data. The certificateless cryptosystem has positive aspects of both PKI and ID-based system. In particular, the confidentiality or the validity of the data of users are guaranteed even if the KGC is malicious, because the KGC does not know the secret value generated by the user. In addition, the user can implicitly confirm an owner of the public key without the certificate since the user needs ID as the part of the public key to encrypt/verify the data. Therefore, constructing certificateless cryptographic schemes such as signature schemes is a meaningful work. In this chapter, we discuss an aggregate signature scheme in the certificateless setting.

6.1.2 Contribution

In this chapter, we propose a certificateless aggregate signature scheme. Our scheme is a variant of an escrow-free identity-based signature scheme rather than a pure certificateless signature scheme, and achieves following properties: (1) our scheme achieves Girault's Level-2 security; (3) our scheme is secure against the strongest adversary in the certificateless setting. We describe the details below.

Certificateless cryptosystem does not need a certificate generated by CA to verify a user's public key, so it does not have the certificate management problem suffered in conventional PKI-based cryptosystem. On the other hand, it also solves the key escrow problem suffered in ID-based cryptosystems since a secret key generated by each user is an unknown value for a malicious KGC. The security model used for the analysis of our scheme does not capture an actively malicious KGC who generates a pair of a secret key and its corresponding public key for any user. Namely, our proposed scheme achieves Girault's Level-2 security. However, based on the idea of [98] proposed by Wu et. al. in 2009, it is easy to modify our certificateless signature into a new kind of signature scheme named *certificate-based signature scheme* [50, 60, 98] in which the Girault's Level-3 security can be achieved. But, with this modification, the public key PK_{ID} of an entity ID will not be able to update at any time without any assistance from KGC whereas this is possible in our scheme. Therefore, here we only discuss how to protect a certificateless signature scheme under the assumption that a secret value

[1] Several researchers avoid to view ID as public key in ID-based system since ID is not a randomly generated value as set in the traditional PKI. Even so, ID-based system can be judged as an answer to the question, "Is it possible to construct a public-key system with a fixed-value public key?" and we describe ID as a part of the public key.

in either PKI or ID-based cryptosystem is kept secret and that malicious activities of KGC are restricted not to fake a pair of secret and public keys described above, i.e. security level-2. In other words, our scheme can resist signature forgery unless KGC impersonates a target signer by generating a key of the target signer.

Since public keys in certificateless cryptosystems are not certified by trusted authorities, these public keys can be replaced by an adversary [44, 63]. According to Huang et al. [44], there are three types of the adversary, normal, strong and super. The normal adversary cannot obtain signatures of a target signer once he/she replaces the public key of the target signer. The strong adversary can obtain signatures of the target signer by providing a secret value corresponding to the replaced public key for a challenger in the security model described in section 6.2.2. The super adversary can also obtain signatures of the target signer but without providing the secret value for the challenger. During the attack, the super adversary can replace a public key pk_A of a target signer Alice with a public key pk_B of another target signer Bob while such an attack cannot be performed by the strong adversary that cannot compute a secret key corresponding to pk_B. This means that the super adversary can access Alice as a black box knowledge extractor for the secret value of Bob without being detected by Bob, because in this scenario signatures, which are output of Alice, are computed from the secret value of Bob. Namely, the adversary trying to forge a signature of the target signer can obtain secret-key related information without being detected by the target signer. In this sense, the super adversary can be judged as the strongest adversary among three types of adversaries.

6.1.3 Application

As applications of certificateless cryptosystem, we focus on ID-federation services for cloud computing. The main motivation of certificateless cryptosystem is for a situation that (1) ID-based cryptosystem is suitable and (2) the insider threat is an inherent problem. As the existing applications of authentication for cloud computing, Yang et al. [101] suggested to combine an ID-federation service such as OpenIDConnect [75] and identity-based cryptography. In the combining system, each user can utilize an identity-based scheme, with an ID information registered in the ID-federation service, for authentication between multi-domain clouds. However, as described above identity-based cryptosystem has the key escrow problem as the inherent problem. In the system in [101], each cloud has an individual key generation center and this center generates a secret key for the ID information.

The key escrow problem means that these key generation centers may be malicious, and thus certificateless cryptosystem is more suitable for this system. Meanwhile, since certificateless cryptosystems require a key including a random number, each user has to provide any information such a subject key identifier. Hence, we also note that certificateless cryptosystem is unsuitable for applications such as S-BGP where total size of data must be decreased.

6.1.4 Related Work

To the best of our knowledge, the existing certificateless aggregate signature schemes have been proposed in [21, 39, 99, 106, 107, 105]. The security of schemes in [21, 39] have never been proven against the super adversary [44]. The security proof in [99] is wrong and this scheme becomes insecure against the super adversary. The schemes in [106, 107, 105] are secure against the super adversary, but only the scheme in [105] achieves the fixed size of signatures. However, in the scheme [105] guarantee of relation between user's identity and its public key seems to be weak, and we consider that the scheme is insufficient in the sense that users may be confused for distinguishing public keys.

6.2 Certificateless Aggregate Signature Scheme

In this section, we define a syntax of certificateless aggregate signature scheme.

6.2.1 Syntax

A certificateless sequential aggregate signature scheme consists of following six algorithms.

Setup Given a security parameter 1^k as input, generate a public parameter $param$, a master secret key msk and its corresponding public key mpk. Return $param$, mpk and msk.

User-Key-Gen Given $param$ as input, generate a user secret key usk_i and its corresponding user public key upk_i. Return usk_i and upk_i.

Partial-Private-Key-Extract Given $param$ and msk and an identity ID_i as input, generate a partial private key d_i. Return d_i.

Set-Key Given $param, mpk, ID_i, d_i, usk_i$ and upk_i, generate a full secret key sk_i, and as a corresponding public key pk_i. Return sk_i and pk_i for ID_i.

Signing Given $param, mpk, sk_i, pk_i, m_i$, and ID_i as input, generate a signature σ_i and then return σ_i.

Aggregation Given $param, mpk, \{ID_j\}_{j=1}^{i}, \{pk_j\}_{j=1}^{i}, \{m_j\}_{j=1}^{i}$ and $\{\sigma_j\}_{j=1}^{i}$, return an aggregate signature σ on $\{m_j\}_{j=1}^{i}$ for users $\{ID_j\}_{j=1}^{i}$.

Verification Given $param, mpk, \{ID_j\}_{j=1}^{i}, \{pk_j\}_{j=1}^{i}, \{m_j\}_{j=1}^{i}$ and an aggregate signature σ as input, check that σ_i is a valid signature on $\{m_j\}_{j=1,\cdots,i}$. If not, return $reject$. Otherwise, return $accept$.

6.2.2 Security Model

In this section, we define a security model of a certificateless aggregate signature scheme. Our security model is constructed by applying a notion of super-adversary in [44] to the security model for sequential aggregate signature scheme in [66].

For certificateless signature scheme, we have to discuss two following types of adversaries with different ability. In the security games, a challenger $\mathcal{C}$ and each adversary who can access a random oracle exist as entities.

Type 1 This type of adversary, $\mathcal{A}_1$, is a dishonest user who does not have the master secret key msk but can replace a public key upk_i of any user ID_i with any chosen value.

Type 2 This type of adversary, $\mathcal{A}_2$, is a malicious KGC who has msk but cannot replace a public key of a target signer.

Definition of Oracles

In the security game in this chapter, we define the following oracles. We denote by $x^{(j)}$ j-th query to access the oracles for all x. Here, $\mathcal{C}$ has a certificate list $\mathcal{L}$ to register users' information.

Create-User Given an identity ID_i, if ID_i has already been queried, nothing will be output. Otherwise, run the algorithms **User-Key-Gen** and **Partial-Private-Key-Extract**, and generate a user secret key usk_i,

a corresponding user public key upk_i and a partial private keyd_i. Run **Set-Key**, and then register (ID_i, pk_i) in $\mathcal{L}$. Return pk_i. In this case, we say that ID_i is created.

Public-Key-Replace Given ID_i and pk'_i chosen by an adversary, if ID_i has already been created, the original public key for ID_i is replaced with pk'_i and re-register (ID_i, pk'_i) in $\mathcal{L}$. Otherwise, return $\perp$.

Secret-Value-Extract Given ID_i, if ID_i has already been created, output a user secret key usk_i corresponding to an original user public key upk_i. Otherwise, return $\perp$. This oracle does not return the user secret key corresponding to the replaced public key pk'_i.

Partial-Private-Key-Extract Given ID_i, if ID_i has already been created, return a partial private value d_i for ID_i. Otherwise, return $\perp$.

Sign Given (ID_i, pk_i, m_i), if ID_i has already been created, return a signature σ_i by **Signing**. Otherwise, return $\perp$. Here pk_i may be either the original public key generated by ID_i or a public key replaced by the adversary [2].

Game 1

This game is executed between $\mathcal{C}$ and $\mathcal{A}_1$.

Setup-Phase $\mathcal{C}$ runs **Setup** algorithm to obtain $param, msk$ and mpk. $\mathcal{C}$ runs $\mathcal{A}_1$ with input $param$ and mpk.

Queries $\mathcal{A}_1$ can access all the oracles described in section 6.2.2 and obtains the outputs from $\mathcal{C}$.

Forgery $\mathcal{A}_1$ outputs a forgery $(\{ID^*_j\}_{j=1,\cdots,n}, \{m^*_j\}_{j=1,\cdots,n}, \sigma^*_n)$ and checks that the following conditions hold.

- σ^*_n is a valid signature on $\{m^*_j\}_{j=1,\cdots,n}$ under $\{pk^*_j\}_{j=1,\cdots,n}$ for $\{ID^*_j\}_{j=1,\cdots,n}$.

[2] In the normal adversary, pk_i is required to be the original pubic key by ID_i. On the other hand, in the strong adversary, if pk_i is replaced, then the corresponding secret key sk_i is required as the additional input. In this paper, by the ability of the super adversary, the signing oracle requires only pk_i even if pk_i is replaced.

- Exactly one $ID^*_{i^*}$ who has never been queried for **Partial-Private-Key-Extract** oracle exists.
- Each ID^*_i in $\{ID^*_j\}_{j=1,\cdots,n}$ does not appear more than once.
- For $ID^*_{i^*}$, $(m^*_{i^*}, ID^*_{i^*}) \notin \{(m^{(h)}_{i^*}, ID^*_{i^*})\}^{(q_s)}_{h=1}$ holds.

$\mathcal{C}$ outputs *accept* if all the conditions hold. Otherwise, $\mathcal{C}$ outputs *reject*.

Definition 21. $\mathcal{A}_1$ breaks a certificateless aggregate signature scheme with $(\epsilon, q_c, q_r, q_s, q_p, q_h, q_{sig}, n, t)$ if $\mathcal{C}$ outputs *accept* in the above game with a success probability greater than ϵ within the execution time t, where $\mathcal{A}_1$ who does not know msk can generate at most q_c create-user queries, q_r public-key-replace queries, q_s secret-value-extract queries, q_p partial-private-key-extract queries, q_h random oracle queries and q_{sig} signing queries, and n is the number of signers included in the forgery.

Game 2

This game is executed between $\mathcal{C}$ and $\mathcal{A}_2$.

Setup $\mathcal{C}$ runs **Setup** algorithm to obtain $param, msk$ and mpk. $\mathcal{C}$ runs $\mathcal{A}_2$ with input $param, msk$ and mpk.

Queries $\mathcal{A}_2$ can access all the oracles described in section 6.2.2 and obtains the outputs.

Forgery $\mathcal{A}_2$ outputs a forgery $(\{ID^*_j\}_{j=1,\cdots,n}, \{m^*_j\}_{j=1,\cdots,n}, \sigma^*_n)$ and check that the following conditions hold.

- σ^*_n is a valid signature on $\{m^*_j\}_{j=1,\cdots,n}$ in ψ^*_n under $\{pk^*_j\}_{j=1,\cdots,n}$.
- Exactly one $ID^*_{i^*}$ who has never been queried for secret-value-extract oracle and public-key-replace oracle.
- Each ID^*_i does not appear more than once in $\{ID^*_j\}_{j=1,\cdots,n}$.
- For $ID^*_{i^*}$, $(m^*_{i^*}, ID^*_{i^*}) \notin \{(m^{(h)}_{i^*}, ID^*_{i^*}\}^{(q_s)}_{h=1}\}$ holds.

$\mathcal{C}$ outputs *accept* if all the conditions hold. Otherwise, $\mathcal{C}$ outputs *reject*.

Definition 22. $\mathcal{A}_2$ breaks a certificateless aggregate signature scheme with $(\epsilon, q_c, q_r, q_s, q_h, q_{sig}, n, t)$ if $\mathcal{C}$ outputs *accept* in the above game with a success probability greater than ϵ within the execution time t, where $\mathcal{A}_2$ can generate at most q_c create-user queries, q_r public-key-replace queries, q_s secret-value-extract queries, q_h random oracle queries and q_{sig} signing queries, and n is the number of signers.

6.2.3 Technical Problem for Constructing Certificateless Aggregate Signatures Secure against Super Adversaries

Technical difficulty for constructing certificateless is due to guarantee of a relation between a user and its public key. This difficulty notably affects a security proof against the type 1 adversary. The adversary can replace a public key and do not have to provide a secret key corresponding to the replaced public key. Essentially, the security against the type 1 adversary who is a dishonest user can be guaranteed as long as the adversary does not know a partial private key d_i. In such a proof, a reduction algorithm need to simulate the partial-private-key-extract oracle the signing oracle without msk. If the adversary replace a public key of a target signer in this situation, the reduction algorithm has to generate signatures of the target signer without not only msk_i but also sk_i. Such a simulation is difficult and complicated, and the success probability becomes quite low. One might think that the reduction algorithm can return the signatures by fully separating a partial private key d_i and a user secret key usk_i and generating each signature individually, but such an approach does not allow users to guarantee relations with their public keys. This means that there is no advantage of utilizing use's ID information.

6.3 Proposed Scheme

In this section, we propose a certificateless sequential aggregate signature scheme. First, we briefly describe our approach to prove the security and then describe the construction.

6.3.1 Our Approach

Our approach is to input a concatenation of a user's ID and a user public key upk_i as input of **Partial-Private-Key-Extract** algorithm. This has a good effect on a relation between ID and the public key. Meanwhile, a reduction algorithm can simulate signatures even if a public key pk is

replaced, because a part of ID and a user public key can be individually simulated. In addition, even if an adversary can know a partial private key d_i, the adversary cannot obtain a full secret key sk_i. In order to execute this procedure, each user has to generate a pair of usk_i and upk_i before the key generation center generate d_i. In the existing certificateless signature schemes, such an ordering of the procedure is not considered and individually generated keys are combined in **Set-Key**. By ordering these algorithms, we can prove the security against the super adversaries. There is an escrow-free ID-based signature scheme [101] as a similar approach, and our approach can be also considered as an extension to a scheme for multiple signers.

6.3.2 Construction

In our scheme, we use state information s similarly with the paper [36]. The state information is one-time information such as time-stamp, and is used to efficiently aggregate the data size of signatures according to [36]. In our scheme, *Signing* phase is run by each signer in turn, and the signature is implicitly aggregated in *Signing* phase instead of an aggregate phase in papers [17, 36, 105].

Setup Generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \mathrm{e})$, and generate a generator $g \leftarrow \mathbb{G}$ and a random number $a \leftarrow \mathbb{Z}_p^*$. Then set $A = g^a$, and choose hash functions $H_1 : \{0,1\}^* \times \{0,1\}^{\rightarrow}\mathbb{G}$, $H_2 : \{0,1\}^* \rightarrow \mathbb{G}$, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}$, and $H_4 : \{0,1\} \rightarrow \mathbb{Z}_p^*$. Finally, return $(p, \mathbb{G}, \mathbb{G}_T, \mathrm{e}, g, H_1, H_2, H_3, H_4)$ as $param$, A as mpk and a as msk.

User-Key-Gen Given $param$, generate a random number $t_i \leftarrow \mathbb{Z}_p^*$ and computes $T_i = g^{t_i}$. Then return t_i as a user secret key and T_i as a corresponding user public key.

Partial-Private-Key-Extract Given $(param, msk, mpk, ID_i \parallel T_i)$, compute $g_{i,j} = H_1(ID_i \parallel T_i, j)$ for $j = \{0,1\}$ and then compute $g_{i,j}^a$. Return $g_{i,j}^a$, $j = \{0,1\}$, as a a partial private key d_i for ID_i.

Set-Key Given $(param, mpk, ID_i, d_i, usk_i, upk_i)$, set $(g_{i,0}^a, g_{i,1}^a, t_i)$ as a full secret key sk_i and (ID_i, T_i) as its corresponding full public key pk_i.

Signing Given $(param, mpk, m_i, ID_i, sk_i)$, generate a string s that it has never used before, and compute $V = H_2(s)$, $W_i = H_3(s \parallel m_i \parallel ID_i \parallel T_i)$,

$c_i = H_4(s \parallel m_i \parallel ID_i \parallel T_i)$. Generate a random number $r_i \leftarrow \mathbb{Z}_p^*$, and then compute as follows:

$$S_i = V^{r_i} g_{i,0}^a \left(g_{i,1}^a\right)^{c_i} W_i^{t_i}, \tag{6.1}$$

$$R_i = g^{r_i} \cdot R_{i-1}. \tag{6.2}$$

Return $\sigma_i = (S_i, R_i, s)$.

Aggregation Given $(param, mpk, \{ID_j\}_{j=1}^i, \{pk_j\}_{j=1}^i \{m_j\}_{j=1}^i, \{\sigma_j\}_{j=1}^i)$, compute as follows:

$$S = \prod_{j=1}^{i} S_j, \tag{6.3}$$

$$R = \prod_{j=1}^{r} R_j. \tag{6.4}$$

Return $\sigma = (S, R, s)$.

Verification Given $(param, mpk, \{ID_j\}_{j=1}^i, \{pk_j\}_{j=1}^i, \{m_j\}_{j=1}^i, \sigma$, parse σ as (S, R, s) and then verify that the following equation holds:

$$\mathrm{e}(S, g) \stackrel{?}{=} \mathrm{e}(V, R) \cdot \mathrm{e}\left(\prod_{j=1}^{i} g_{j,0} g_{j,1}^{c_j}, A\right) \cdot \prod_{j=1}^{i} \mathrm{e}\left(W_j, T_j\right), \tag{6.5}$$

where, for all j, $g_{j,l} = H_1(ID_j \parallel T_j, l)$ for $l = \{0, 1\}$, $V = H_2(s)$, $W_j = H_3(s \parallel m_j \parallel ID_j \parallel T_j)$, $c_j = H_4(s \parallel m_j \parallel ID_j \parallel T_j)$.

6.3.3 Correctness

From the signature equations , the following equation holds:

$$\begin{aligned}
\mathrm{e}\left(S, g\right) &= \mathrm{e}\left(V^r \prod_{j=1}^{i} \left(g_{j,1}^a (g_{j,2}^a)^{c_j} W_j^{t_j}\right), g\right) \\
&= \mathrm{e}\left(V, g^r\right) \cdot \mathrm{e}\left(\prod_{j=1}^{i} g_{j,1} (g_{j,2})^{c_j}, g^a\right) \prod_{j=1}^{i} \mathrm{e}\left(W_j, g^{t_j}\right) \\
&= \mathrm{e}\left(V, R\right) \cdot \mathrm{e}\left(\prod_{j=1}^{i} g_{j,1} (g_{j,2})^{c_j}, A\right) \prod_{j=1}^{n} \mathrm{e}\left(W_j, T_j\right).
\end{aligned}$$

6.3.4 Security Analysis

In this section, we discuss the security of the proposed scheme against adversaries described in Section 6.2.2. In particular, when the adversary breaks the proposed scheme in each game, we construct an algorithm $\mathcal{B}$ to solve CDH problem by using the adversary.

Theorem 23. The proposed scheme is secure against type 1 of the adversary with $(\epsilon, q_c, q_r, q_s, q_p, q_{h_1}, q_{h_2}, q_{h_3}, q_{h_4}, q_{sig}, n, t)$ if (t', ϵ')-CDH assumption holds, where

$$\begin{aligned} \epsilon' &= \left(\epsilon - \frac{q_{sig}(q_{sig}-1)}{2p}\right) \frac{27}{(q_p + q_{h_1} + q_{h_4} + (q_{h_1} + q_{h_2} + q_{h_4})q_{sig})^3} \cdot \frac{1}{e^3}, \\ t' &= t + t_e(2q_{h_1} + q_{h_2} + q_{h_4} + q_c + 2q_p + 6q_{sig} + 3n + 1), \end{aligned}$$

where t_e is the computational time for a single exponentiation.

Proof. This proof is based on the security proof in paper [36], and we define a probability δ to set 1 for tossing a coin. To complete the proof, we finally determine a concrete value of δ.

Given a CDH challenge $(g, g^a, g^b, p, \mathbb{G}, \mathbb{G}_T, \mathrm{e})$, $\mathcal{B}$ who tries to solve CDH problem sets $mpk = g^a$ and a certification list $\mathcal{L} = \emptyset$. This means that $\mathcal{B}$ implicitly sets a as msk. Then $\mathcal{B}$ sets ID-list $[\cdot,\cdot]$ H_1-list $[\cdot,\cdot,\cdot,\cdot,\cdot,\cdot]$, H_2-list $[\cdot,\cdot,\cdot]$, H_3-list $[\cdot,\cdot]$ and H_4-list $[\cdot,\cdot,\cdot,\cdot,\cdot]$ as empty, and run $\mathcal{A}$ with g, g^a as input. Here, without loss of generality, we assume that $\mathcal{B}$ executes H_1-query and H_2-query before executing H_3-query and H_4-query, H_1-query before executing the create-user query and each random oracle query before executing the signing oracle query.

H_1-query Given a string $ID_i \parallel T_i$ generated by $\mathcal{A}$, check that H_1-list includes ID_i. If so, return $H_1(ID_i, j)$ from H_1-list, where $j = 0, 1$. Otherwise, toss a coin $H_1\text{-}coin_i \leftarrow \{0,1\}$ with probability δ. If $H_1\text{-}coin_i = 0$, generate $\alpha_{i,0}, \alpha_{i,1} \leftarrow \mathbb{Z}_p$ and set $\alpha'_{i,0} = \alpha'_{i,1} = 0$. Otherwise, generate $\alpha_{i,0}, \alpha_{i,1}, \alpha'_{i,0}, \alpha'_{i,1} \leftarrow \mathbb{Z}_p$. Set $H_1(ID_i, j) = (g^{\alpha_{i,j}}(g^b)^{\alpha'_{i,j}})$, and register $(ID_i \parallel T_i, H_1\text{-}coin_i, \alpha_{i,0}, \alpha_{i,1}, \alpha'_{i,0}, \alpha'_{i,1})$ on H_1-list. Return $H_1(ID_i, j)$ for $j = \{0, 1\}$.

H_2-query Given s generated by $\mathcal{A}$, check that H_2-list includes s. If so, return $H_2(s)$ from H_2-list. Otherwise, toss a coin $H_2\text{-}coin_k \leftarrow \{0,1\}$ and generate $\beta \leftarrow \mathbb{Z}_p^*$. If $H_2\text{-}coin_k = 0$, set $V = (g^b)^\beta$ as $H_2(s)$. Otherwise, set $V = g^\beta$ as $(H_2(s)$. Register $(s, H_2\text{-}coin_k, \beta)$ on H_2-list and return $H_2(s)$.

H_3-query Given $s \parallel m_i \parallel ID_i \parallel pk_i$ generated by $\mathcal{A}$, check that H_3-list includes $s \parallel m_i \parallel ID_i \parallel pk_i$. If so, return $H_3(s \parallel m_i \parallel ID_i \parallel pk_i)$ from H_3-list. Otherwise, generate $\gamma \leftarrow \mathbb{Z}_p^*$ and set $H_3(s \parallel m_i \parallel ID_i \parallel pk_i) = g^\gamma$. Register $(s \parallel m_i \parallel ID_i \parallel pk_i, \gamma)$ on H_3-list and return $H_3(s \parallel \psi_i)$.

H_4-query Given $s \parallel m_i \parallel ID_i \parallel pk_i$ generated by $\mathcal{A}$, check that H_4-list includes $s \parallel m_i \parallel ID_i \parallel pk_i$. If so, return $H_4(s \parallel m_i \parallel ID_i \parallel pk_i)$ from H_4-list. Otherwise, toss a coin $H_4\text{-}coin_l \leftarrow \{0,1\}$. If $H_4\text{-}coin_l = 0$, check that $H_1\text{-}coin_i = H_2\text{-}coin_k = 1$ for $s \parallel L_i$. If so, check that $s \parallel L_i \neq s \parallel L_i'$ exists with $ID_i = ID_i'$. If so, aborts. Otherwise, set $H_4(s \parallel m_i \parallel ID_i \parallel pk_i) = -\frac{\alpha_{i,0}'}{\alpha_{i,1}'}$. Otherwise, generate $d_{(i,k,l)} \leftarrow \mathbb{Z}_p^*$. Set $H_4(s \parallel m_i \parallel ID_i \parallel pk_i) = d_{(i,k,l)}$. Register $(s, ID_i, m_i \parallel ID_i \parallel pk_i, H_4\text{-}coin_l, d_{(i,k,l)})$ on H_4-list and return $H_4(s \parallel m_i \parallel ID_i \parallel pk_i)$.

Create-User Given ID_i generated by $\mathcal{A}$, check that $\mathcal{L}$ includes ID_i. If so, return (ID_i, T_i) from $\mathcal{L}$. Otherwise, generate $t_i \leftarrow \mathbb{Z}_p$ and set $T_i = g^{t_i}$. Then, retrieve $H_1(ID_i \parallel T_i, j)$ for $j = \{0,1\}$ from H_1-list as $g_{i,j}$, and register (ID_i, T_i) in $\mathcal{L}$ and (ID_i, t_i) in ID-list. Return $H_1(ID_i, j)$ for $j = \{0,1\}$ and T_i.

Partial-Private-Key-Extract Given ID_i generated by $\mathcal{A}$, check that $\mathcal{L}$ includes ID_i. If not, nothing will be output. Otherwise, check that $H_1\text{-}coin_i = 1$ holds. If so, abort. Otherwise, set $g_{i,j}^a = (g^a)^{\alpha_{i,j}}$ and return $g_{i,j}^a$ where $j = 0, 1$.

Public-Key-Replace Given ID_i and T_i' generated by $\mathcal{A}$, re-register (ID_i, T_i') in $\mathcal{L}$ and $(ID_i, \perp)$, where $\perp$ means an error symbol.

Secret-Value-Extract Given ID_i generated by $\mathcal{A}$, check that $\mathcal{L}$ includes ID_i. If not, nothing will be output. Otherwise, return t_i from $\mathcal{L}$. Here, if the secret value corresponding to ID_i in ID-list is nil, then nothing will be output.

Signing Given $\{m_j\}_{j=1,\cdots,i}, \{ID_j\}_{j=1,\cdots,i}, \psi_i, \sigma_i, s$ generated by $\mathcal{A}$, check that $H_1\text{-}coin_i$, $H_2\text{-}coin_k$ and $H_4\text{-}coin_l$. If $H_1\text{-}coin_i = H_2\text{-}coin_k = H_4\text{-}coin_l = 1$, abort. Otherwise, compute a signature as follows. In the case that $H_1\text{-}coin_i = 0$, generate a random number $r \leftarrow \mathbb{Z}_p^*$ and pick the latest public key T_i of ID_I from $\mathcal{L}$, which may be the original public key generated from

Create-User or a false public key replaced by the adversary. Then, compute as follows:

$$S_i = V^r \left(g^a\right)^{\alpha_{i,0}} \left(g^a\right)^{\alpha_{i,1}c_i} \left(T_i\right)^{\gamma}, \tag{6.6}$$
$$R_i = g^r, \tag{6.7}$$

where V, γ and c_i are retrieved from $H_2\text{-}list, H_3\text{-}list$ and $H_4\text{-}list$. These values become a valid signature. In the case that $H_1\text{-}coin_i = 1 \wedge H_2\text{-}coin_i = 0$, compute as follows:

$$S_i = \left(\left(g^b\right)^{\beta}\right)^r \left(g^a\right)^{\alpha_{i,0}} \left(g^a\right)^{\alpha_{i,1}c_i} \left(T_i\right)^{\gamma}, \tag{6.8}$$
$$R_i = g^r \left(g^a\right)^{-\frac{\alpha'_{i,0}+\alpha'_{i,1}c_i}{\beta}}, \tag{6.9}$$

where β, γ and c_i are retrieved from $H_2\text{-}list, H_3\text{-}list$ and $H_4\text{-}list$. These values become a valid signature since they can be written as follows:

$$\begin{aligned} S_i &= \left(\left(g^b\right)^{\beta}\right)^r \left(g^a\right)^{\alpha_{i,0}} \left(g^a\right)^{\alpha_{i,1}c_i} \left(T_i\right)^{\gamma} \cdot \frac{\left(\left(g^b\right)^a\right)^{\alpha'_{i,0}+\alpha'_{i,1}c_i}}{\left(\left(g^b\right)^a\right)^{\alpha'_{i,0}+\alpha'_{i,1}c_i}} \\ &= \left(\left(g^b\right)^{\beta}\right)^{r-a\frac{\alpha'_{i,0}+\alpha'_{i,1}c_i}{\beta}} \left(g^{\alpha_{i,0}}\left(g^b\right)^{\alpha'_{i,0}}\right)^a \left(g^{\alpha_{i,1}}\left(g^b\right)^{\alpha'_{i,1}}\right)^{ac_i} W_i^{t_i}, \end{aligned} \tag{6.10}$$
$$R_i = g^{r-a\frac{\alpha'_{i,0}+\alpha'_{i,1}c_i}{\beta}}. \tag{6.11}$$

In the case that $H_1\text{-}coin_i = H_2\text{-}coin_i = 1 \wedge H_4\text{-}coin_i = 0$, compute as follows:

$$S_i = \left(g^{\beta}\right)^r \left(g^a\right)^{\alpha_{i,0}} \left(g^a\right)^{\alpha_{i,1}\left(-\frac{\alpha'_{i,0}}{\alpha'_{i,1}}\right)} \left(T_i\right)^{\gamma}, \tag{6.12}$$
$$R_i = g^r. \tag{6.13}$$

These values become a valid signature since they can be written as follows:

$$\begin{aligned} S_i &= \left(g^{\beta}\right)^r \left(g^a\right)^{\alpha_{i,0}} \left(g^a\right)^{\alpha_{i,1}\left(-\frac{\alpha'_{i,0}}{\alpha'_{i,1}}\right)} \left(T_i\right)^{\gamma} \left(\left(g^b\right)^{\alpha'_{i,0}}\right)^a \left(\left(g^b\right)^{\alpha'_{i,1}\left(-\frac{\alpha'_{i,0}}{\alpha'_{i,1}}\right)}\right)^a \\ &= V^r \left(g^{\alpha_{i,0}}\left(g^b\right)^{\alpha'_{i,0}}\right)^a \left(g^{\alpha_{i,1}}\left(g^b\right)^{\alpha'_{i,1}}\right)^{a\left(-\frac{\alpha'_{i,0}}{\alpha'_{i,1}}\right)} W_i^{t_i}, \end{aligned} \tag{6.14}$$
$$R_i = g^r. \tag{6.15}$$

Output Given a forgery $(\{ID_j^*\}_{j=1,\cdots,n}, \{pk_j^*\}_{j=1,\cdots,n}, \{m_j^*\}_{j=1,\cdots,n}, \sigma^*)$ output by $\mathcal{A}$ after iterations of the above queries, check that $H_1\text{-}coin_i = H_2\text{-}coin_i = H_4\text{-}coin_i = 1$ holds. If not, abort. Otherwise, $\mathcal{B}$ can extract the solution of CDH problem. Here, the forgery can be written as $S^* = V^r \prod_{i=1}^n \left(g_{i,0}^a (g_{i,1}^a)^{c_i}\right) \prod_{i=1}^n W_i^{t_i}$, $R^* = g^r$ since these are a valid signature. $\mathcal{B}$ computes as follows:

$$g'^a = \left(\frac{\frac{S^*}{(R^*)^\beta \left(\prod_{j=1 \wedge j \neq i^*}^n T_i^\gamma\right)}}{\left(\prod_{j=1}^n (g^a)^{\alpha_{i,0}} (g^a)^{\alpha_{i,1} c_i}\right)} \right)^{\frac{1}{\alpha'_{i,0} + \alpha'_{i,1} c_i}} \tag{6.16}$$

Since $\mathcal{B}$ knows all the values, $\mathcal{B}$ can compute the above equation. The probability ϵ' that $\mathcal{B}$ solves can be obtained as follows:

$$\begin{aligned} \epsilon' &= \Pr[forge \wedge \overline{abort} \wedge \overline{collide}] \\ &= \Pr[\overline{abort}] \cdot \left(\Pr[forge|\overline{abort}] - \Pr[collide|\overline{abort}]\right), \end{aligned}$$

where $forge$ means an event that $\mathcal{A}$ succeeds in breaking the scheme, *collide* means an event that $\mathcal{A}$ outputs $(\{m_j^*\}_{j,\cdots,n}, \psi_n^*, \sigma_n^*)$ such that it has previously been queried to *Signing* oracle and *abort* means an event that $\mathcal{B}$ aborts the simulation with $\mathcal{A}$. $\Pr[forge|\overline{abort}] = \epsilon$ holds from definition of the adversary and, from birthday paradox, $\Pr[collide|\overline{abort}]$ can be obtained as follows:

$$\Pr[collide|\overline{abort}] = \frac{q_{sig}(q_{sig} - 1)}{2p} \tag{6.17}$$

In addition, $\Pr[\overline{abort}]$ can be written as follows:

$$\Pr[\overline{abort}] = \Pr[\overline{abort_p} \wedge \overline{abort_{h_4}} \wedge \overline{abort_{sig}} \wedge \overline{after}], \tag{6.18}$$

where $abort_p$ means the event that $\mathcal{B}$ aborts the simulation with $\mathcal{A}$ for partial-private-key-extract query. Similarly, We denote by $abort_{h_4}$ an event for H_4 query and by $abort_{sig}$ one for signing query. Here $abort_x$ means the event that $\mathcal{B}$ aborts the simulation with $\mathcal{A}$ during the x-query, where $x \in \{p, h_4, sig\}$ and each p, h_4, sig stands for partial-private-key-extract, H_4 and signing, respectively. $after$ means that $\mathcal{B}$ aborts after $\mathcal{A}$ outputs the forgery. Each event can be written as follows:

$$\Pr[\overline{abort_p}] = (1-\delta)^{q_p}, \tag{6.19}$$

$$\Pr[\overline{abort_{h_4}}] = (1-\delta)^{q_{h_1} + q_{h_2}}, \tag{6.20}$$

$$\Pr[\overline{abort_{sig}}] = (1-\delta)^{(q_{h_1} + q_{h_2} + q_{h_4}) q_{sig}}, \tag{6.21}$$

$$\Pr[\overline{after}] = \delta^3, \tag{6.22}$$

where δ is a probability that $\mathcal{B}$ tosses 1 for the coin tosses. To complete the proof, we give the maximum value for δ. Let $f(\delta)$ be the following function.

$$\begin{aligned} f(\delta) &= (1-\delta)^{q_p}\,(1-\delta)^{q_{h_1}+q_{h_2}}\,(1-\delta)^{(q_{h_1}+q_{h_2}+q_{h_4})q_{sig}}\,\delta^3 \\ &= (1-\delta)^{q_p+q_{h_1}+q_{h_2}+(q_{h_1}+q_{h_2}+q_{h_4})q_{sig}}\,\delta^3. \end{aligned} \tag{6.23}$$

To be easily written, we denote $a = q_p + q_{h_1} + q_{h_2} + (q_{h_1} + q_{h_2} + q_{h_4})q_{sig}$. From the derived function, f is maximized at $\delta_{max} = \frac{3}{a}$. Here, $f(\delta_{max})$ can be written as follows:

$$f(\delta_{max}) = \frac{27}{a^3}\left(1-\frac{3}{a}\right)^a. \tag{6.24}$$

From definition of base of natural logarithm, we can compute as follows:

$$\lim_{a\to\infty}\left(1-\frac{3}{a}\right)^a = \frac{1}{e^3}, \tag{6.25}$$

$$\begin{aligned} \therefore \epsilon' &= \left(\epsilon - \frac{q_{sig}(q_{sig}-1)}{2p}\right) f_1(\delta_{opt}) \\ &= \left(\epsilon - \frac{q_{sig}(q_{sig}-1)}{2p}\right)\cdot\frac{27}{a^3}\cdot\frac{1}{e^3}. \end{aligned} \tag{6.26}$$

The execution time of $\mathcal{B}$ is the execution time of $\mathcal{A}$ plus two exponentiations for H_1-**Query**, one exponentiation for H_2-**Query**, one exponentiation for H_4-**Query**, one exponentiation for **Create-User** queries, two exponentiations for **Partial-Private-Key-Extract** queries, at most six exponentiations for **Signing** queries, and $3n+1$ exponentiations in the final step. Therefore,

$$t' = t + t_e(2q_{h_1} + q_{h_2} + q_{h_4} + q_c + 2q_p + 6q_{sig} + 3n + 1), \tag{6.27}$$

where t_e is a computational time for a single exponentiation. □

Theorem 24. The proposed scheme is secure against type 2 of the adversary with $(\epsilon, q_c, q_s, q_{h_1}, q_{h_2}, q_{h_3}, q_{h_4}, q_{sig}, n, t)$ if and only if (t', ϵ')-CDH assumption holds, where

$$\epsilon' = \left(\epsilon - \frac{q_{sig}(q_{sig}-1)}{2p}\right)\frac{27}{(q_s + (q_{h_1}+q_{h_2}+q_{h_3})q_{sig})^3}\cdot\frac{1}{e^3}, \tag{6.28}$$

$$t' = t + t_e(2q_{h_1} + q_{h_2} + q_{h_4} + q_c + 6q_{sig} + 3n + 2), \tag{6.29}$$

and t_e is the computational time for the final result.

Proof. This proof is based on the security proof in paper [100], and we also define a probability δ to set 1 for tossing a coin. To complete the proof, we finally determine a concrete value of δ.

In this proof, we assume that $\mathcal{B}$ executes H_1-query and H_2-query before executing H_3-query and H_4-query, H_1-query before executing the create-user query and each random oracle query before executing the signing oracle query. Given a CDH challenge value $(g, g^a, g^b, p, \mathbb{G}, \mathbb{G}_T, \mathrm{e})$, $\mathcal{B}$ who tries to solve CDH problem generates $x \leftarrow \mathbb{Z}_p^*$ as msk, and sets $A = g^x$ as mpk and a certification list $\mathcal{L} = \emptyset$. Then $\mathcal{B}$ sets ID-list $[\cdot, \cdot, \cdot]$ H_1-list $[\cdot, \cdot, \cdot]$, H_2-list $[\cdot, \cdot, \cdot]$, H_3-list $[\cdot, \cdot, \cdot, \cdot]$ and H_4-list $[\cdot, \cdot]$ as empty, and run $\mathcal{A}$ with g, x, A as input.

H_1-query Given $ID_i \parallel T_i$ generated by $\mathcal{A}$, check that H_1-list includes ID_i. If so, return $H_1(ID_i, j)$ from H_1-list where $j = \{0, 1\}$. Otherwise, generate $(\alpha_{i,0}, \alpha_{i,1}) \leftarrow \mathbb{Z}_p$, and then set $H_1(ID_i, j) = g^{\alpha_{i,j}}$ for $j = \{0, 1\}$. Register $(ID_i \parallel T_i, \alpha_{i,0}, \alpha_{i,1})$ on H_1-list, and return $H_1(ID_i \parallel T_i, j)$ for $j = \{0, 1\}$.

H_2-query This execution is exactly the same as Game 1.

H_3-query Given $s \parallel m_i \parallel ID_i \parallel pk_i$ generated by $\mathcal{A}$, check that H_3-*list* includes $s \parallel m_i \parallel ID_i \parallel pk_i$. If so, return $H_3(s \parallel m_i \parallel ID_i \parallel pk_i)$ from H_3-*list*. Otherwise, generate $\gamma \leftarrow \mathbb{Z}_p^*$ and toss a coin $H_3\text{-}coin_l \leftarrow \{0, 1\}$ with the probability δ. If $H_3\text{-}coin_l = 0$, set $H_3(s \parallel m_i \parallel ID_i \parallel pk_i) = g^\gamma$. Otherwise, $H_3(s \parallel L_i) = (g \cdot (g^b))^\gamma$. Register $(ID_i, s_i \parallel m_i \parallel ID_i \parallel pk_i, H_3\text{-}coin_l, \gamma)$ on H_3-list and return $H_3(s \parallel m_i \parallel ID_i \parallel pk_i)$.

H_4-query Given $s \parallel m_i \parallel ID_i \parallel pk_i$ generated by $\mathcal{A}$, check that H_4-*list* includes $s \parallel m_i \parallel ID_i \parallel pk_i$. If so, return $H_4(s \parallel m_i \parallel ID_i \parallel pk_i)$ from H_4-*list*. Otherwise, generate $d_{(i,k,l)} \leftarrow \mathbb{Z}_p^*$ and set $H_4(s \parallel m_i \parallel ID_i \parallel pk_i) = d_{(i,k,l)}$. Register $(s \parallel m_i \parallel ID_i \parallel pk_i, d_{(i,k,l)})$ on H_4-list and return $H_4(s \parallel m_i \parallel ID_i \parallel pk_i)$.

Create-User Given ID_i generated by $\mathcal{A}$, check that $\mathcal{L}$ includes ID_i. If so, return (ID_i, T_i) from $\mathcal{L}$. Otherwise, toss a coin $ID\text{-}coin_i \leftarrow \{0, 1\}$ with probability δ and generate $t_i \leftarrow \mathbb{Z}_p$. If $ID\text{-}coin_i = 0$, set $T_i = g^{t_i}$. Otherwise, set $T_i = (g^a)^{t_i}$. Then, retrieve $H_i(ID_i \parallel T_i, j)$ for $j = \{0, 1\}$ from H_1-list as $g_{i,j}$, Register (ID_i, T_i) in $\mathcal{L}$ and -register $(ID_i, ID\text{-}coin_i, t_i)$ in ID-list. Return $H_1(ID_i \parallel, j)$ and T_i as ID_i's public key pk_i.

Secret-Value-Extract Given ID_i generated by $\mathcal{A}$, check that $\mathcal{L}$ include ID_i. If not, nothing will be output. Otherwise, check that $ID\text{-}coin_i = 1$ holds. If so, abort. Otherwise, return t_i.

Signing Given a signing query (m_i, ID_i, pk_i, s) generated by $\mathcal{A}$, check that $ID\text{-}coin_i$, $H_2\text{-}coin_k$ and $H_3\text{-}coin_l$ in the query with each list. If $ID\text{-}coin_i = H_2\text{-}coin_k = H_3\text{-}coin_l = 1$, abort. Otherwise, generate a random number $r \leftarrow \mathbb{Z}_p^*$ and generate a signature as follows. In the case that $H_3\text{-}coin_l = 0$, compute as follows:

$$S_i = V^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (T_i)^\gamma, \tag{6.30}$$

$$R_i = g^r, \tag{6.31}$$

where V, γ and c_i are retrieved from $H_2\text{-}list, H_3\text{-}list$ and $H_4\text{-}list$. These values become a valid signature since the following equation holds:

$$\begin{aligned} S_i &= V^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (T_i)^\gamma = V^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (g^{x_i})^\gamma \\ &= V^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (W_i)^{x_i}, \end{aligned} \tag{6.32}$$

where x_i is a secret key corresponding to a public key T_i. In the case that $H_3\text{-}coin_i = 1 \wedge H_2\text{-}coin_i = 0$, compute as follows:

$$S_i = \left((g^b)^\beta\right)^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (T_i)^\gamma, \tag{6.33}$$

$$R_i = g^r (T_i)^{-\frac{\gamma}{\beta}}, \tag{6.34}$$

These values also become a valid signature since the following equation holds:

$$\begin{aligned} S_i &= \left((g^b)^\beta\right)^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (T_i)^\gamma \\ &= \left((g^b)^\beta\right)^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (g^{x_i})^\gamma (g^b)^{x_i\gamma - x_i\gamma} \\ &= \left((g^b)^\beta\right)^r g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (g^{x_i}(g-b)^{x_i})^\gamma (g^b)^{-x_i\gamma} \\ &= \left((g^b)^\beta\right)^{r-\frac{x_i\gamma}{\beta}} g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} \left(g \cdot g^b\right)^{x_i\gamma} \\ &= \left((g^b)^\beta\right)^{r-\frac{x_i\gamma}{\beta}} g_{i,0}^x \left(g_{i,1}^x\right)^{c_i} (W_i)^{x_i}, \end{aligned} \tag{6.35}$$

$$R_i = g^r \cdot (g^{x_i})^{-\frac{\gamma}{\beta}} = g^{r-\frac{x_i\gamma}{\beta}}. \tag{6.36}$$

Here, we write $r' = r - \frac{\gamma x_i}{\beta}$. Then, the following equations can be written:

$$S_i = (V)^{r'} g_{i,0}^b \left(g_{i,1}^b\right)^{c_i} (W_i)^{x_i}, \tag{6.37}$$

$$R_i = g^{r'}. \tag{6.38}$$

Output Given a forgery $(\{ID_i^*\}_{i=1}^n, \{m_i^*\}_{i=1}^n, \{pk_i^*\}_{i=1}^n, \sigma_n^*)$ output by $\mathcal{A}$, check that $ID\text{-}coin_i = H_2\text{-}coin_i = H_3\text{-}coin_i = 1$ holds. If not, abort. Otherwise, similarly with Game 1, $\mathcal{B}$ can extract the solution of CDH problem from the forgery as follows:

$$g^{ab} = \left(\frac{\frac{S^*}{(R^*)^\beta \left(\prod_{j=1 \wedge j \neq i^*}^n T_j^\gamma\right)}}{(g^a)^{t_{i^*}\gamma} \left(\prod_{j=1}^n g_{j,0}^b (g_{j,1}^b)^{c_i}\right)} \right)^{\frac{1}{t_{j^*}\gamma}}. \tag{6.39}$$

Since $\mathcal{B}$ knows all the values, $\mathcal{B}$ can compute the above equation. Similarly with Game 1, we can compute the maximum value for $f(\delta)$ and the computational time for $\mathcal{B}$.

$$\begin{aligned} \epsilon' &= \Pr[forge \wedge \overline{abort} \wedge \overline{collide}] \\ &= \Pr[\overline{abort}] \cdot \left(\Pr[forge|\overline{abort}] - \Pr[collide|\overline{abort}]\right). \end{aligned}$$

Here, $\Pr[forge|\overline{abort}] = \epsilon$ and $\Pr[collide|\overline{abort}] = \frac{q_{sig}(q_{sig}-1)}{2p}$ hold similarly with Game 1. In addition, $\Pr[\overline{abort}]$ can be written as follows:

$$\Pr[\overline{abort}] = \Pr[\overline{abort_s} \wedge \overline{abort_{h_3}} \wedge \overline{abort_{sig}} \wedge \overline{after}], \tag{6.40}$$

where $abort_s$ means the event that $\mathcal{B}$ aborts the simulation with $\mathcal{A}$ for **Secret-Value-Extract**. Each event can be written as follows:

$$\Pr[\overline{abort_s}] = (1-\delta)^{q_s}, \tag{6.41}$$

$$\Pr[\overline{abort_{sig}}] = (1-\delta)^{(q_c+q_{h_2}+q_{h_3})q_{sig}}, \tag{6.42}$$

$$\Pr[\overline{after}] = \delta^3, \tag{6.43}$$

where δ is a probability that $\mathcal{B}$ tosses 1 for its coin tosses. Similarly with Game 1, we give the maximum value for δ. We define $f(\delta)$ as the following function.

$$\begin{aligned} f(\delta) &= (1-\delta)^{q_s} (1-\delta)^{(q_c+q_{h_2}+q_{h_3})q_{sig}} \delta^3 \\ &= (1-\delta)^{q_s+(q_c+q_{h_2}+q_{h_s})q_{sig}} \delta^3. \end{aligned} \tag{6.44}$$

Here, we denote $X = q_s + (q_{h_1} + q_{h_2} + q_{h_4})q_{sig}$ for short. From the derived function, f is maximized at $\delta_{max} = \frac{3}{X}$. Therefore, similarly with Game 1, the following equation can be obtained.

$$\begin{aligned} \epsilon' &= \left(\epsilon - \frac{q_{sig}(q_{sig}-1)}{2p}\right) f_1(\delta_{max}) \\ &= \left(\epsilon - \frac{q_{sig}(q_{sig}-1)}{2p}\right) \cdot \frac{3}{X^3} \cdot \frac{1}{e^3}. \end{aligned} \tag{6.45}$$

The execution time of $\mathcal{B}$ can be also obtained similarly with Game 1. The execution time of $\mathcal{B}$ is the execution time of $\mathcal{A}$ plus two exponentiations for H_1-**Query**, one exponentiation for H_2-**Query**, one exponentiation for H_4-**Query**, one exponentiation for **Create-User** queries, at most six exponentiations for **Signing** queries, and $3n + 2$ exponentiations in the final step. Therefore,

$$t' = t + t_e(2q_{h_1} + q_{h_2} + q_{h_4} + q_c + 6q_{sig} + 3n + 2), \tag{6.46}$$

where t_e is a computational time in the final step. □

6.4 Construction Resisting the DoD Attack

Liu et al.[63] have pointed out a problem in distributing public keys in a certificateless setting. Suppose an adversary replace a public key of any user with other faked public key. Then an encryptor who cannot detect the replacement, certificateless property, performs the encryption under the faked public key. Such data encrypted under the faked public key cannot be decrypted by the user correctly because the user does not know a secret value corresponding to the replaced faked public key. This attack is called Denial of Decryption (DoD) attack. In order to prevent this attack, they have proposed a method to guarantee the validity of a public key without the interaction with any trusted authority, i.e. *self-generated-certificate*. In this method, each user guarantees the validity of a public key by generating a certificate, signature, under a secret key corresponding to the public key.

DoD attack may also occur in digital signature scheme in that a digital signature generated by any user is maliciously rejected by the replacement of its own public key. In this approach, Wu proposed a digital signature scheme with self-generated-certificate[97]. Since the user can detect the replacement of the public key by the verification with the self-generated-certificate, it can resist against malicious rejection of signature. However, the construction with the self-generated-certificate cannot achieve level-3 security. In particular, the malicious KGC can still impersonate any user by generating a pair of a secret key and a public key and its corresponding self-generated-certificate by him-/herself.

The notion of self-generated-certificate can be applied to our scheme. In paper [103], which is a previous version of this work, we proposed a CLOSAS scheme with self-generated-certificate. In this section, we give the detail of the construction. Although the following construction cannot be achieved

level-3 security, the proposed scheme becomes more secure in the sense that the scheme resist DoD attack.

Construction

Setup This algorithm is same as the scheme in section 6.3.2.

User-Key-Gen Given $param$, generate random numbers $(t_{i,0}, t_{i,1}) \leftarrow \mathbb{Z}_p^*$ and computes $T_{i,0} = g^{t_{i,0}}$ and $T_{i,1} = g^{t_{i,1}}$. Then return $(t_{i,0}, t_{i,1})$ as a user secret key usk_i and $(T_{i,0}, T_{i,1})$ as a corresponding user public key upk_i.

Partial-Private-Key-Extract Given $(param, msk, mpk, ID_i \parallel T_{i,0} \parallel T_{i,1})$, compute $g_{i,j} = H_1(ID_i \parallel T_{i,0} \parallel T_{i,1}, j)$ for $j = \{0, 1\}$ and then compute $g_{i,j}^a$. Return $g_{i,j}^a$ for $j = \{0, 1\}$ as a partial private key d_i.

Set-Key Given $(param, mpk, ID_i, d_i, usk_i, upk_i)$, set $(g_{i,0}^a, g_{i,1}^a, t_{i,0}, t_{i,1})$ as a full secret key sk_i, and generate a random number r_i' and state information s_i. Then set $m_i' := ID_i \parallel T_{i,1}$ and compute as follows:

$$S_i' = V_i^{r_i'} g_{i,0}^a \left(g_{i,1}^a\right)^{c_i'} W_i'^{t_{i,0}}, \tag{6.47}$$

$$R_i' = g^{r_i'}, \tag{6.48}$$

where $V_i = H_2(s_i)$, $W_i = H_3(s_i \parallel m_i')$ and $c_i' = H_4(s_i \parallel m_i')$. Set $\sigma_i' = (S_i', R_i', s_i)$, and then set $(ID_i, T_{i,0}, T_{i,1}, \sigma_i')$ as a corresponding full public key pk_i. Return (sk_i, pk_i).

Signing Given $(param, mpk, m_i, ID_i, sk_i)$, generate a string s that it has never used before, and compute $V = H_2(s)$, $W_i = H_3(s \parallel m_i \parallel ID_i \parallel T_{i,1})$, $c_i = H_4(s \parallel m_i \parallel ID_i \parallel T_{i,1})$. Generate a random number $r_i \leftarrow \mathbb{Z}_p^*$, and then compute as follows:

$$S_i = V^{r_i} g_{i,0}^a \left(g_{i,1}^a\right)^{c_i} W_i^{t_{i,1}}, \tag{6.49}$$

$$R_i = g^{r_i} \cdot R_{i-1}. \tag{6.50}$$

Return $\sigma_i = (S_i, R_i, s)$.

Aggregation Given $(param, mpk, \{ID_j\}_{j=1}^{i}, \{pk_j\}_{j=1}^{i}\{m_j\}_{j=1}^{i}, \{\sigma_j\}_{j=1}^{i})$, compute as follows:

$$S = \prod_{j=1}^{i} S_j, \tag{6.51}$$

$$R = \prod_{j=1}^{r} R_j. \tag{6.52}$$

Return an aggregate signature $\sigma = (S, R, s)$.

Verification Given $(param, mpk, \{ID_j\}_{j=1}^{i}, \{pk_j\}_{j=1}^{i}, \{m_j\}_{j=1}^{i}, \sigma,$ verify that, for $\{ID_j\}_{j=1}^{i}$, the public key pk_j is correct. In particular, parse the signers' self-generated-certificates σ'_j in pk_j as (S'_j, R'_j, s_j) for $j = [1, n]$, and set $m'_j := ID_j \parallel T_{j,1}$. Then, for $j = [1, n]$, check if the following equation holds:

$$\mathrm{e}(S'_j, g) \stackrel{?}{=} \mathrm{e}(V_j, R'_j) \cdot \mathrm{e}\left(g_{j,0} g_{j,1}^{c'_j}, A\right) \cdot \mathrm{e}\left(W'_j, T_{j,0}\right), \tag{6.53}$$

where, for all j, $g_{j,l} = H_1(ID_j \parallel T_{j,1}, l)$ for $l = \{0, 1\}$, $V_j = H_2(s_j)$, $W'_j = H_3(s_j \parallel m'_j)$ and $c'_j = H_4(s_j \parallel m'_j)$. If not, output *reject*. Otherwise, parse σ as (S, R, s) and then verify that the following equation holds:

$$\mathrm{e}(S, g) \stackrel{?}{=} \mathrm{e}(V, R) \cdot \mathrm{e}\left(\prod_{j=1}^{i} g_{j,0} g_{j,1}^{c_j}, A\right) \cdot \prod_{j=1}^{i} \mathrm{e}\left(W_j, T_{j,1}\right), \tag{6.54}$$

where, for all j, $V = H_2(s)$, $W_j = H_3(s \parallel m_i \parallel ID_i \parallel T_{i,1})$, and $c_j = H_4(s \parallel m_i \parallel ID_i \parallel T_{i,1})$. If the above equation holds, output *accept*. Otherwise, output *reject*.

Theorem 25. A self-generated-certificate in the proposed scheme is existentially unforgeable if there is no adversary who breaks the proposed scheme described in Section 6.3.2.

Proof (Sketch). Intuitively, if an adversary who can forge a self-generated-certificate exists, then the adversary can also forge an aggregate signature in the proposed scheme in this paper by using the forged self-generated-certificate as a signature of the target signer. This result conflicts with the theorems described in the previous section. □

Table 6.1: Evaluation of the schemes: For the evaluation of the signing cost and the verification cost, we denote by $\mathcal{P}$ the computational cost of pairing, by $\mathcal{H}$ the computational cost of a map-to-point functions, by ε the computational cost of a single exponentiation, and by $\mathcal{L}(p)$ the binary length of p. The relation with public key means a relation between ID and a corresponding public key.

Schemes	Computational Cost for Signer	Computational Cost for Verifier	Signature Size	Relation with Public Key
ZQWZ10 [105]	$4\varepsilon + 5\mathcal{H}$	$5\mathcal{P} + 2n\varepsilon$ $+(4n+3)\mathcal{H}$	$2\mathcal{L}(p)$	Weak
Proposed Scheme	$4\varepsilon + 2\mathcal{H}$	$(3+n)\mathcal{P} + n\varepsilon$ $+(3n+1)\mathcal{H}$	$2\mathcal{L}(p)$	Strong

6.5 Evaluation

We compare the performance of the proposed scheme with some existing schemes with respect to the signing cost, the verification cost, the signature size, type of the scheme and certificateless property. The result is shown in table 6.1.

As shown in Table 6.1, our scheme has the same signature size as the ZQWZ10 scheme [105]. In terms of the computational costs, our proposed scheme is efficient in the signing cost, and the number of the map-to-point function and the number of the exponentiations become less. Hence, our scheme becomes quite faster over a curve, which is suitable in the bilinear maps but is unsuitable in the map-to-point function and the exponentiations, e.g., type A curve, which provides a symmetric pairing, in PBC library [65]. In addition, our scheme guarantees a stronger relation between user's ID and its public key.

Chapter 7

Unrestricted Sequential Aggregate Signatures

7.1 Introduction

7.1.1 Background

Overview *Aggregate signature* [17] allows each signer to generate a signature for an individual message and the generated signatures can be aggregated later by anyone. If the aggregation of signatures is performed by the signing algorithm at the same time as generating a new signature (especially, if the aggregation of signatures needs the signing key), then such a scheme is called *sequential aggregate signature* [66]. Besides its interesting mathematical structure, sequential aggregate signature also has practical importance among aggregate signatures because of several potential applications such as certificate chains in hierarchical public-key infrastructures. However, most of the existing (sequential) aggregate signature schemes have a restriction that a message or a signer cannot appear more than once during a sequential signing process, which is not desirable in some applications described in Section 7.1.3. To overcome the problem, Bellare et al. introduced the notion of *unrestricted aggregate signature* [9] which is an aggregate signature without the above-mentioned restriction. According to [17], a new problem arises for proving the security in the unrestricted setting. To the authors' best knowledge, all the existing unrestricted schemes without random oracles require multilinear maps in order to prove the security [41, 82], and consequently, these schemes are impractical.Therefore, it is an important problem from both theoretical and practical viewpoints to realize unrestricted

sequential aggregate signature (in the standard model) without multilinear maps, which is the motivation of our present work.

Plain Public Key Model vs. Certified Key Model Signature schemes with multiple signers such as aggregate signature schemes require special care for the rogue key attack [67]. The rogue key attack is an attack that a malicious signer chooses its public key as a function of that of honest signers in such a way that it can then easily forge a signature in a signing group. By utilizing such a key, rogue key attackers can easily forge signatures. Interestingly, the rogue key attackers do not know a secret key corresponding to the rogue public key, because the value includes secret keys of victims. Hence, one of possible way to prevent the rogue key attack is to utilize a zero-knowledge proof of the secret key for a key registration [67]. The model requiring a user to prove knowledge of its secret key during public key registration with a certificate authority (CA) is called the *certified key model* [13]. Bellare and Neven [10] pointed out a gap in which existing public key infrastructures (PKI) do not include such a protocol, and suggested the *plain public key model* where key registration with a CA ensures nothing about a party's possession or knowledge of a secret key. A security proof via the plain public key model guarantees the strongest security for aggregate signature, and the scheme in [17] required the restricted setting in order to prove the security in the plain public key model until Bellare et al. proved the security in the unrestricted setting [9]. However, its proof is done in the random oracle model and it is an open problem whether there is an unrestricted sequential aggregate signature scheme in the plain public key model, which is proven secure in the standard model. On the other hand, if we admit the use of widely recognized protocols such as the Schnorr Identification [84] or the Fischlin paradigm [34], the rogue key attack can be prevented. With the use of such protocols, we only need to discuss the certified key model. As in the plain public key model, it is not known whether there is an unrestricted sequential aggregate signature scheme even in a weaker security model, the certified key model, which is proven secure without multilinear maps in the standard model. Therefore, in this paper, we adopt the certified key model and study under the model a secure unrestricted sequential aggregate signature scheme in the standard model. In the course of the study, we identify that several schemes, which are secure in the restricted setting, become insecure in the unrestricted setting even in the certified key model.

Limitation of Straightforward Approach Roughly speaking, the difficulty to realize unrestricted sequential aggregate signature is that, when a signer generates many signatures in a signing process, the *same* secret key corresponding to the signer is used many times. Therefore, a "trivial" approach is to use a *new* secret key for each signing operation to generate and aggregate a new signature (cf., [1]). This approach essentially imitates a situation that all signers are distinct. However, even for such a "trivial" construction, there is a restriction that the signer cannot generate signatures more than the number of keys. In the paper, we aim at realizing by a different approach a scheme that does not have such a restriction and can deal with any (polynomially bounded) number of signatures.

7.1.2 Our Contributions

We show that unrestricted sequential aggregate signature which is provably secure in the standard model can be realized without multilinear maps. More precisely, we revisit the Waters-hash-based sequential aggregate signature scheme proposed by Lu et al. [64], the LOSSW06 scheme for short, which does not use multilinear maps. We note that the original security proof for the LOSSW06 scheme highly relies on a restriction that each message or signer appears at most once, therefore the unrestricted property described in Section 7.1.1 cannot be achieved by straightforward extensions of the original proof (see Section 7.3 for details). We develop a new technique to prove the security even in the presence of multiple messages and signers, which reveals that the LOSSW06 scheme is in fact an unrestricted sequential aggregate signature scheme. To the authors' best knowledge, this is the first result to provide an unrestricted sequential aggregate signature without multilinear maps which is secure in the standard model.

Roughly speaking, our new proof technique implies the following: For a sequential aggregate signature scheme in the standard model, if the hash function used in the scheme can be chosen as a programmable hash function and the scheme has a certain re-randomization functionality, then the scheme is an unrestricted sequential aggregate signature in the standard model. In particular, the LOSSW06 scheme satisfies the condition, therefore the above-mentioned result is derived. We note that the reduction cost of a security proof for unrestricted sequential aggregate signature by our new technique is almost the same as that for sequential aggregate signature by a standard proof technique. Moreover, as another example of applications of our new proof technique, we also revisit an ID-based sequential aggregate signature scheme proposed by Boldyreva et al. [15], the BGOY10 scheme for short, in the random oracle model, and show that the scheme is also an

unrestricted sequential aggregate signature scheme (we note that random oracles can be seen as "ideal" programmable hash functions [40]). To the authors' best knowledge, the BGOY10 scheme is the first ID-based sequential aggregate signature scheme in the unrestricted setting. We also note that the LOSSW06 scheme and the BGOY10 scheme can be extended to the framework of ordered multisignature, which leads us to the notion of unrestricted ordered multisignature via our results in the paper.

We emphasize that, in contrast to the case of the two schemes above, it is *not always* true that sequential aggregate signature scheme is also unrestricted sequential aggregate signature. Actually, we show that a scheme by Lee et al. [54], which is secure in the sense of sequential aggregate signature, falls into insecure when multiple messages and signers are in addition allowed (we note that we do *not* claim that a flaw exists in the original security proof in [54]).

7.1.3 Applications

For practical use of sequential aggregate signature, the unrestricted property for multiple messages/signers is getting more effective as each signer generates signatures more frequently. For such cases, when the scheme lacks the unrestricted property, a signer needs all the data previously generated by the signer for generating and aggregating a new signature, which is obviously too inefficient. In contrast, for the unrestricted schemes, by utilizing the Lazy Verification technique [18], it becomes possible to update the previously aggregated signatures by using the differential data only. Hence the efficiency is significantly improved, especially, for example, in the following applications:

Content Editing Systems Here we discuss content marketing based on content-editing systems [83], where aggregate signature authorizes each re-editing of an already uploaded content such as a movie on YouTube [104]. In ordinary systems, when the same user re-edits a content and then generates a signature again, the user has to remove the previous signature before the editing and to generate a signature for the whole content after the editing, which requires the user to possess (even temporarily) the whole content. Since such a content is usually very large, the process of manipulating the signatures is too memory-inefficient [71]. On the other hand, for such systems using unrestricted sequential aggregate signature, the cost of updating a signature at each re-editing is much reduced owing to the above-mentioned property that only the difference data suffice to update

Table 7.1: **Comparison of Known Sequential Aggregate Signature Schemes.** Here the two cells written in the bold fonts are due to our results in the paper. In the column of type of scheme, we denote by Synchronized a synchronized scheme, by General a general aggregate signature scheme, by IB an ID-based signature scheme and by Sequential a sequential aggregate signature scheme.

Schemes	Type of Scheme	Provable Secure in Unrestricted Model?	Proof Model	Multilinear Maps
AGH10 [1]	Synchronized	No	Standard	No
BGLS03 [17]	General	Yes	ROM	No
HW13 [41]	General	Yes	Standard	Yes
LLY13-1 [54]	Sequential	No	Standard	No
LLY13-2 [55]	Sequential	No	Standard	No
LMRS04 [66]	Sequential	Yes	ROM	No
Neven08 [71]	Sequential	Yes	ROM	No
RS09 [82]	General	Yes	Standard	Yes
Schröder [85]	Sequential	No	Standard	No
LOSSW06 [64]	Sequential	**Yes**	Standard	No
BJ10 [6]	IB,General	Yes	ROM	No
GR06 [36]	IB, Syncronized	No	ROM	No
GLOW12 [37]	IB, Sequential	No	ROM	No
HW13 [41]	IB, General	Yes	Standard	Yes
BGOY10 [15]	IB, Sequential	**Yes**	ROM	No

the signature. Moreover, as a side effect derived from the property of sequential aggregate signature that only the signer (knowing the secret key) can aggregate the generated signature, combining individual contents with valid signatures by a third party can be prevented in comparison with unrestricted general aggregate signature.

Privacy-Preserving Electronic Toll Pricing Systems Privacy-preserving electronic toll pricing (PrETP) system [7] is a system that a user, who is an on-board unit of a vehicle, can prove that they use genuine data and perform correct operations while disclosing the minimum amount of time/location data. Informally speaking, when the vehicle passes a gate on a road, a service provider sends the time/location data with the provider's signature to the on-board unit, and then the on-board unit of the vehicle generates a commitment for the received data. A main advantage of utilizing unrestricted

sequential aggregate signature is a compression of the signatures sent by the same provider. It was discussed in [7] that, practically, each on-board unit has only a small memory (e.g., 100 kilobytes) while it has to store all the received data during a somewhat long time period (e.g., one month). Here we focus on the issue of storing all the signatures associated to these data, which was not intensively mentioned in [7] but would in fact be a serious problem in practical situations. Unrestricted sequential aggregate signature is expected to reduce the cost of on-board unit's aggregating the signatures generated by the same provider.

7.1.4 Comparison with Priori Works

Comparison with the Dividing-Out Method : To deal with multiple appearance of messages/signers by some sequential aggregate signature schemes, it was proposed in [64] to remove the previous signature and then re-generate a signature at each time to update a signature (called the divide-out method). However, as already mentioned in [71], such a method is not reasonable for some applications described in Section 7.1.3, since the method requires each signer to possess the whole of previously used data to update a signature. As discussed in Section 7.1.3, the data size in the content editing systems may be too large, and the overloads of the memory become heavy. In addition, in the PrETP system, each user has to submit to the service provider all of the path data in order to compress the data. This results in reveal of the user's privacy data to the provider. On the other hand, by using unrestricted sequential aggregate signature schemes presented by our result in the paper (combined with the Lazy Verification technique [18]), each update of a signature requires the new data only as mentioned in Section 7.1.3, therefore the usefulness is significantly improved in comparison with the previous technique. We describe the detail in Section 7.3.3.

Unrestricted Aggregation Without Random Oracles : The first aggregate signature scheme was proposed by Boneh et al. [17], and the signing operation and the aggregation operation were separated. Such aggregate signature schemes called the general aggregate signature, and signatures in these schemes can be aggregated into a single signature by anyone. Meanwhile, Lysyanskaya et al. [66] proposed the first sequential aggregate signature scheme where the aggregation operation is executed at the same time as the signing operation. In other words, the aggregation operation cannot be individually executed without the secret key. Although there are many aggregate signature schemes, these schemes have a restriction that all

messages as well as all signers have to be distinct. The known applications of aggregate signature is secure-border gateway protocol [52] or certificate chains. Messages to be signed in these systems are IP address or certificate information which are unique information, and signers correspond to routers of network service provider or certificate authority. Since these entities do not appear more than once, the restriction has been ignored in the existing works. Bellare et al. [9] formalized the problem as unrestricted aggregate signature, and a main application of unrestricted aggregate signature is sensor networks. Neven [71] proposed an unrestricted sequential aggregate signature scheme with message recovery from permutations, and his scheme was proposed in the random oracle model. In the standard model, Rückert and Schröder [82] and Hohenberger and Waters [41] proposed unrestricted general aggregate signature schemes with multilinear maps. While the Rückert-Schröder scheme [82] was proposed in the certified key model [13], the Hohenberger-Waters scheme [41] was proposed in the plain public key model [10]. The certified key model is a model assuming that each signer knows a secret key corresponding to an own public key, and is the most discussed and practical model for a signature scheme with multiple signers [1, 13, 14, 54, 55, 64, 85]. To the best of our knowledge, there is no unrestricted aggregate signature scheme without the multilinear maps even in the certified key model. A main difference between the certified key model and the plain public key model is whether an adversary is allowed to execute the rogue key attack or not. The plain public key model allows the adversary to execute the rogue key attack, and thus the strongest security can be guaranteed in this model. However, generally speaking, the rogue key attack can be prevented by a zero-knowledge proof of secret information [67].

Unrestricted ID-based Sequential Aggregation : ID-based aggregate signature [36] is aggregate signatures in ID-based cryptography [86]. The first aggregate signature scheme by Gentry and Ramzan [36] was proposed in the synchronized setting [36, 1] where each signer signs only once per period. The scheme in the synchronized setting does not allow each signer to appear multiple times for an aggregate signature. Although there are schemes in [6, 41] which are secure without such an assumption, these schemes have some problems for a practical scenario. In the scheme in [6] signers themselves need to be designated in the signing algorithm, and a signature of a new signer cannot be added in an aggregate signature. On the other hand, the scheme in [41] is based on the multilinear maps, whose cost is too much large. In addition to these problems, the schemes in [6, 41]

are not sequential aggregate signature scheme but general aggregate signature scheme. ID-based sequential aggregate signature was formalized by Boldyreva et al. [14]. Although they also proposed an ID-based sequential aggregate signature scheme, their scheme was broken by Hwang et al. [45]. After, they revised the scheme [15], and their new scheme, the BGOY10 scheme, is provably secure in the restricted setting under a newly proposed interactive assumption described in Chapter 2. We extend the BGOY10 scheme to an unrestricted ID-based sequential aggregate signature scheme. More recently, Gerbush et al. [37] improved the security reduction of the BGOY10 scheme, called the GLOW12 scheme, by the dual-system methodology [96], and hence the security of the GLOW12 scheme is reduced to a static assumption in the restricted setting. A main difference between the BGOY10 scheme and the GLOW12 scheme is the security assumptions, but these algebraic structures are essentially identical. Thus, we expect that the security of the GLOW12 scheme can be proven in the unrestricted setting. Providing such a proof is an open problem.

7.2 Unrestricted Sequential Aggregate Signature Scheme

We define a syntax of unrestricted sequential aggregate signature and its security model. These are defined in the [64]. Whereas the LOSSW06 model [64] treats with only the restricted setting, unrestricted sequential aggregate signature can be treated as discussed in detail below. We describe the definitions in conjunction with the notion of lazy verification [18] in order for an efficiency discussion as described in Section 7.3.3.

7.2.1 Syntax

Sequential aggregate signature scheme consists of the following algorithms.

Setup(1^k) Given a security parameter 1^k, return a public parameter $para$.

KeyGeneration$(para)$ Given $para$, return a secret key sk_i and its corresponding public key pk_i.

Signing$(sk_i, m_i, \boxed{\{m_j\}_{j=1}^{i-1}, \{pk_j\}_{j=1}^{i-1},} \sigma_{i-1}, pk_i)$ Given sk_i, a message m_i to be signed, a set $\{m_j\}_{j=1}^{i-1}$ of signed messages, a set $\{pk_j\}_{j=1}^{i-1}$ of public keys, an aggregate signature σ_{i-1} and pk_i, return a new aggregate signature σ_i on $\{m_j\}_{j=1}^{i}$ under $\{pk_j\}_{j=1}^{i}$.

Verification$(\{m_j\}_{j=1}^{i}, \sigma_n, \{pk_j\}_{j=1}^{i})$ Given a set $\{m_j\}_{j=1}^{i}$ of signed messages, an aggregate signature σ_i and a set $\{pk_j\}_{j=1}^{i}$ of public keys, return *accept* or *reject*.

Definition 26 (Correctness)**.** In a sequential aggregate signature scheme, we say that the scheme is correct if, for all $para, sk_i$ and pk_i given by **Setup** and **KeyGeneration**, **Verification**$(\{m_j\}_{j=1}^{i}$, **Signing**$(sk_i, m, \{m_j\}_{j=1}^{i-1}$, $\{pk_j\}_{j=1}^{i-1}, \sigma_{i-1}, pk_i)$, $\{pk_j\}_{j=1}^{i-1})$ outputs *accept* for all $i \in [1, n]$.

We note the efficiency and the sequential aggregation as important properties of sequential aggregate signature. In a discussion of the efficiency, the size of an aggregate signature σ_i should be as small as possible relatively to the previously generated one $\sigma i - 1$, i.e., $|\sigma_i| = |\sigma_{i-1}|$ holds. For the sequential aggregation, one of main applications of sequential aggregate signature is certificate chains, and an ability to combine preexisting individual signatures into an aggregate signature is unnecessary. More precisely, each signer transforms a sequential aggregate signature into another that includes a signature on a message of his choice, and the signing and the aggregation are a single operation. Namely, only a signer who knows a secret key can aggregate the signatures.

Lazy verification construction [18] is a construction removing the framed text in **Signing**. The correctness holds even for adopting lazy verification. Strictly speaking, lazy verification is an individual capability from unrestricted sequential aggregate signature, but as described in Section 7.3.3 the applications become more practical by combining these techniques.

7.2.2 Security Model

In this paper, we follow the security model that was proposed by Lu et al. [64]. We note that the security of an unrestricted sequential aggregate signature scheme can be guaranteed by an analysis under the following model. The model is a variant of the certified key model [13], which assumes that each signer knows a secret key corresponding to its own public key. As described in Section 7.1, the following model is different from the model described in [9]. A main difference is that the following model does not allow an adversary to execute rogue key attack while an adversary in the model in [9] is allowed. However, in a high-level discussion, the rogue key attack can be overcome utilizing a zero-knowledge proof [67] in general with respect to a secret key corresponding to its own public key. In a majority of the existing schemes, a public key corresponding to a secret key x is g^x, which

is a famous form. Hence, a knowledge of the secret key can be easily guaranteed by the Schnorr identification scheme [84]. In fact, as a related work, Rückert and Schröder proposed an unrestricted general aggregate signature scheme under the certified key model [82].

There exist an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ in this model. The challenger $\mathcal{C}$ has a certified-key list $\mathcal{L}$ to register users and their own public keys, including and $\mathcal{A}$ can get to know all the keys in $\mathcal{L}$ except for the one given by $\mathcal{C}$ to the target signer. The advantage of $\mathcal{A}$ can be obtained with the probability that $\mathcal{C}$ outputs *accept* in the subsequent game. Hereinafter, we denote by $x^{(i)}$ the value of the i-th query for all x.

Initial Phase The challenger $\mathcal{C}$ generates a public parameter $para$ by **Setup** and a pair of challenge key (sk^*, pk^*) of a target signer by **KeyGeneration**. Then, $\mathcal{C}$ initializes $\mathcal{L} := pk^*$, and runs $\mathcal{A}$ with $para$ and pk^* as input.

Certification Query $\mathcal{A}$ generates sk_i and its corresponding public key pk_i for any signer. Then, $\mathcal{A}$ provides (sk_i, pk_i) to $\mathcal{C}$, and $\mathcal{C}$ registers pk_i in $\mathcal{L}$.

Signing Query For all i, $\mathcal{A}$ generates a signing query $(m^{(h)}, \{m_j\}_{j=1}^{i-1}, \{pk_j\}_{j=1}^{i-1}, \sigma_{i-1}, pk^*)$ as hth query for pk^*, where the following conditions hold for the query: **Verification** algorithm outputs *accept*; For all pk_j, pk_j in $\{pk_j\}_{j-1}^{i-1}$ is included in $\mathcal{L}$; $i-1 < n$ holds. Given such a query by $\mathcal{A}$, $\mathcal{C}$ runs **Signing**$(sk^*, m_i^{(h)}, \{m_j\}_{j=1}^{i-1}, \{pk_j\}_{j=1}^{i-1}, \sigma_{i-1}, pk^*)$, and obtains σ_i. Finally, $\mathcal{C}$ returns σ_i on $\{m_j^{(h)}\}_{j=1}^{i}$ under $\{pk_j\}_{j=1}^{i}$.

Output After q_c iterations of the certification queries and q_s iterations of the signing queries, $\mathcal{A}$ outputs a forgery $(\{m_i^*\}_{i=1}^n, \sigma_n^*)$. Here, the following conditions hold for the forgery where let an index of the position of the target signer be i^*: **Verification**$(\{m_i^*\}_{i=1}^n, \sigma_n^*, \{pk_i^*\}_{i=1}^n)$ outputs *accept*; there exists a set Δ^* of indexes corresponding to pk^* such that $\exists j \in \Delta^*[m_j^* \notin \{m_{i^*}^{(h)}\}_{h=1}^{q_s}]$ and $|\Delta^*| > 0$ holds; $\{pk_i\}_{i=1}^n$ includes pk^*; For all $pk_j \in \{pk_i\}_{i=1}^n$, pk_j is included in $\mathcal{L}$. If all conditions hold, then $\mathcal{C}$ outputs *accept*. Otherwise, $\mathcal{C}$ outputs $reject$.

Definition 27. We say that an aggregate signature scheme is $(t, q_c, q_s, \ell, n, \epsilon)$-secure if there is no adversary $\mathcal{A}$ breaks with $(t, q_c, q_s, \ell, n, \epsilon)$. Here, we say that $\mathcal{A}$ breaks the scheme with $(t, q_c, q_s, \ell, n, \epsilon)$ as that a challenger $\mathcal{C}$ outputs *accept*, in the security game described above, with the probability greater

than ϵ within the execution time t. Here, $\mathcal{A}$ can generate at most q_c certification queries and at most q_s signing queries, ℓ is the length of the message output by $\mathcal{A}$, and n is the number of signers included in the forgery.

Extension to Ordered Multisignatures under Unrestricted Setting

Sequential aggregate signatures give rise to ordered multisignatures where each signer guarantees the validity of a common message and its position in a signing group [15]. A main difference of the ordered multisignatures from the above definition is conditions of queries. Since the ordered multisignatures are multisignatures guaranteeing both the common message and the signing order among the signing group, the queries are given as the form of (m, ψ_i) for all $i \in [1, n]$, where ψ_i is the signing order from the first signer to ith signer, instead of $\{m_i\}_{i=1}^n$. We give the formal definition of unrestricted ordered multisignatures in Section 7.5.

7.2.3 (In)Security of the Existing Schemes under Unrestricted Setting

The model described in the previous section is the most utilized. In this section, we show that there exist schemes which becomes forgeable by multiple appearance of signers: more specifically, we forge signatures in the LLY13 sequential aggregate signature scheme [54] and in the BGOY07 ordered multisignature scheme [14]. As described in Section 7.1.2, we note that we do not claim that there is no flaw in their security proofs. These original proofs are correct, and the following discussion under the unrestricted setting is outside of the scope of their models. We simply claim that a scheme secure under the restricted setting does not imply an unrestricted sequential aggregate signature scheme which is provably secure.

(In)Security of the LLY13 Scheme under Unrestricted Setting

Review of the LLY13 Sequential Aggregate Signature Scheme : We briefly review the LLY13 scheme [54].

Setup(1^k) Generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$. Then, choose random generators $g, Y \in \mathbb{G}$. Output $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, Y)$ as the public parameter $para$.

KeyGeneration$(para)$ Pick random numbers $\alpha_i \leftarrow \mathbb{Z}_p$ and compute $A_i = g^{\alpha_i}$. Output α_i as sk_i and A_i as pk_i.

Signing$(sk_i, m_i, \{m_j\}_{j=1}^{i-1}, \{pk_j\}_{j=1}^{i-1}, \sigma_{i-1}, pk_i)$ Parse the secret key sk_i as α_i, the previous signature σ' as $(S_{i-1}, R_{i-1}, W_{i-1})$ and pk_i as A_i, where if $i = 1$, then set $\sigma' = (1, g, Y)$. Check if **Verification**$(\{m_j\}_{j=1}^{i-1}, \sigma_{i-1}, \{pk_j\}_{j=1}^{i-1})$ outputs *accept*. If not, output $\perp$. Otherwise, pick a random number $r \leftarrow \mathbb{Z}_p$ and compute as $S_i \leftarrow \left(S_{i-1}(R_{i-1})^{\alpha_i}(W_{i-1})^{\alpha_i M_i}\right)^r$, $R_i \leftarrow R_{i-1} \cdot g^r$ and $W_i \leftarrow W_{i-1}$. Output $\sigma_i = (S_i, R_i, W_i)$ as an aggregate signature.

Verification$(\{m_j\}_{j=1}^{i}, \sigma_n, \{pk_j\}_{j=1}^{i})$ Parse σ_i as (S_i, R_i, W_i) and each pk_i as A_i. Check if $\boxed{\text{all of } \{pk_j\}_{j=1}^{i} \text{ are distinct}}$, and output *reject* if not. Otherwise, check if the following equations hold: $\boldsymbol{e}(R_i, Y) \stackrel{?}{=} \boldsymbol{e}(W_i, g)$ and $\boldsymbol{e}(S_i, g) \stackrel{?}{=} \boldsymbol{e}\left(R_i, \prod_{j=1}^{i} A_j\right) \cdot \left(W_i, \prod_{j=1}^{i} V_j^{m_j}\right)$. If the previous equation holds, output *accept*. Otherwise, output *reject*.

The unrestricted setting is a construction removing the framed text in **Verification**. Removing the check that the public keys are distinct is out of the scope of the original construction in the LLY13 scheme [54]. In such a situation, the following attack will occur.

Insecurity under Unrestricted Setting : We show that the LLY13 scheme becomes insecure under the unrestricted setting. The signature equation in the LLY13 scheme can be written as $S = (g^r)^{\sum_{j=1}^{i} \alpha_j}(Y^r)^{\sum_{j=1}^{i} \alpha_j m_j}$ for any i. If the signers are allowed to sign in multiple positions for a single signature, then the attacker can forge a signature as follows: where the signing process is boss||boss: in this situation, the signature equation can be written as $S_2 = (g^r)^{2\alpha_a}(Y^r)^{\alpha_a m_1 + \alpha_a m_2}$, and then any user obtaining the signature can generate another signature as $S_2^{\frac{1}{2}} = (g^r)^{\alpha_a}(Y^r)^{\alpha_a \frac{(m_1+m_2)}{2}}$, which is a value accepted as a signature on $m' := \frac{(m_1+m_2)}{2}$. As described in Section 7.1.2, the proof is correct but this attack is outside the proof. Similarly, a signature $S_1 = (g^r)^{\alpha_a}(Y^r)^{m_1 \alpha_a}$ is convertible into a new one $S_1^2 = (g^r)^{2\alpha_a}(Y^r)^{2m_1\alpha_a}$, which is a forgery on a new message $m' := 2m_1$, under the unrestricted setting.

(In)Security of the BGOY07 Ordered Multisignature Scheme under Unrestricted Setting

This attack is similar as that in the previous section. The construction of the BGOY07 scheme utilizes indexes to represent a position for each signer, and

these indexes can be arbitrarily manipulated by an adversary if any signer appears more than once for a single ordered multisignature. As describe above, their security proof is correct and this attack is outside of the scope of the model in [14]. We show the details of the BGOY07 scheme and the insecurity under the unrestricted setting in Section 7.5.3.

7.3 Construction without Random Oracles

In this section, we show that the LOSSW06 sequential aggregate signature scheme, which was originally proposed as a restricted scheme, is actually an unrestricted scheme in the standard model.

7.3.1 Basic Idea: Why the Original Proof Technique Is Not Enough

Before explaining our idea for the proof, first we note that the extension to the unrestricted setting cannot be achieved by a straightforward generalization of the original security proof in [64] by the following reason. In [64], the security of their proposed scheme in the restricted setting was reduced to the security of the Waters signature, therefore a signing oracle for the Waters signature is available in the proof. However, the oracle can only generate a new signature corresponding to the challenge key, but cannot aggregate the new signature to the previously aggregated signatures, which also requires the unknown challenge key. The original proof resolved this issue by changing the order of the signature generation; the aggregation of the signature corresponding to the challenge key can be skipped by generating the signature first among the sequential signing process. Now the remaining signatures can be successfully aggregated to the former ones, *since the challenge key is not used again by the property of the restricted setting.* In other words, the original proof essentially relies on the assumption that each signer appears at most once; if the challenge key is used twice, then the aggregation of the second signature generated by the challenge key cannot be skipped whatever the order of the signature generation is.

Our proof in the paper resolves the problem by a completely different approach. Namely, instead of reducing the security to that of the Waters signature as in the standard proof technique in the literature, we try to reduce the security of the scheme directly to the CDH assumption, by focusing on the following two properties of the scheme by Lu et al.; (I) the hash function used in the scheme is a kind of programmable hash function; (II) the structure of the scheme admits re-randomization of a signature. The first

step of the proof in the unrestricted setting is the same as the restricted setting discussed above; changing the order of the signature generation in such a way that a number of signatures for the challenge key have to be generated first and then signatures for the other signers are generated. The essential part of the problem is how to generate and aggregate the signatures for the challenge key. The authors found that the property (I) enables us to emulate the sequential generation and aggregation of signatures by *just a single* generation of a signature for the challenge key. Moreover, the property (II) allows us to use the re-randomization of the obtained signature, and the true and the emulated aggregation of signatures can be made *indistinguishable*. These reduce the argument for the unrestricted setting to that for the restricted setting, the latter being already solved. This is an outline of our new proof technique. (We note that, if the Waters signature were equipped with a functionality to emulate the aggregation of signatures as described above, then the original proof in [64] would be also extendible to the unrestricted setting. However, the Waters signature does not satisfy the requirement.)

7.3.2 New Proof of the LOSSW06 Scheme

Through the observation described in the previous section, we prove that the LOSSW06 scheme is a secure unrestricted sequential aggregate signature scheme. The following proof is our main contribution.

Unrestricted LOSSW06 Sequential Aggregate Signature Scheme

In this section, we show the construction of the LOSSW06 scheme under unrestricted setting. The following construction is an improved version via the new security proof. In this scheme, each signer signs an individual document m_i and a message m in this scheme will be dealt as a bit-string of the form $\{0,1\}^\ell$ for all ℓ. We note that the signing order in this scheme only represents a signing group and is not guaranteed by its verifications.

Setup(1^k) Generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ described in Section 2.3. Generate random generators $(g_1, g_2) \in \mathbb{G}$, and output $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g_1, g_2)$ as $para$.

KeyGeneration$(para)$ Choose $\alpha_i, v_i' \leftarrow \mathbb{Z}_p^*$ and ℓ-bit vector $(v_{i,1}, \cdots, v_{i,\ell}) \leftarrow \mathbb{Z}_p^\ell$, and compute $A_i = g_1^{\alpha_i}, V_i' = g_1^{v_i'}, V_{i,1} = g_1^{v_{i,1}}, \cdots, V_{i,\ell} = g_1^{v_{i,\ell}}$. sk_i is $(g_2^{\alpha_i}, v_i', v_{i,1}, \cdots, v_{i,\ell})$, and a corresponding public key pk_i is $(A_i, V_i', V_{i,1}, \cdots, V_{i,\ell})$.

Signing$(sk_i, m_i, \{m_j\}_{j=1}^{i-1}, \{pk_j\}_{j=1}^{i-1}, \sigma_{i-1}, pk_i)$ Parse m_i as a bit-string $(m_{i,1}, \cdots, m_{i,\ell}) \in \{0,1\}^\ell$ and σ_{i-1} as (S', R'). $\boxed{\text{Check if } pk_i \notin \{pk_j\}_{j=1}^{i-1} \text{ holds}}$. Then, generate a random number $r_i \leftarrow \mathbb{Z}_p^*$ and compute as follows:

$$\begin{aligned} R_i &= R_{i-1} \cdot g_1^{r_i}, \\ S_i &= S_{i-1} \cdot g_2^{\alpha_i} (R_{i-1})^{v_i' + \sum_{j=1}^{\ell} v_{i,j} m_{i,j}} \left(\prod_{j=1}^{i} \left(V_j' \prod_{e=1}^{\ell} V_{j,e}^{m_{j,e}} \right) \right)^{r_i}. \end{aligned}$$

Otherwise, compute as follows:

$$\begin{aligned} R_i &= R_{i-1} \cdot, \\ S_i &= S_{i-1} \cdot g_2^{\alpha_i} (R_{i-1})^{v_i' + \sum_{j=1}^{\ell} v_{i,j} m_{i,j}}. \end{aligned}$$

Output $\sigma = (S_i, R_i)$.

Verification$(\{m_i\}_{i=1}^n, \sigma_n, \{pk_i\}_{i=1}^n)$ Parse m_i as a bit-string $(m_{i,1}, \cdots, m_{i,\ell}) \in \{0,1\}^\ell$ for $i = [1, n]$ and σ as (S_n, R_n). Extract each signer's public key $(A_i, V_i', V_{i,1}, \cdots, V_{i,\ell})$ from $\{pk_i\}_{i=1}^n$ and verify that the following equation holds:

$$\boldsymbol{e}(S_n, g_1) \stackrel{?}{=} \boldsymbol{e}\left(g_2, \prod_{i=1}^{n} A_i \right) \cdot \boldsymbol{e}\left(R_n, \prod_{i=1}^{n} \left(V_i' \prod_{j=1}^{\ell} V_{i,j}^{m_{i,j}} \right) \right).$$

Note : The framed text in the signing algorithm and the equation (2) are new computations via the proof under the unrestricted setting. As described in Section 7.3.3, the computational cost can be reduced by these computations in comparison with the divide-out method.

Security Analysis of the LOSSW06 Scheme

The LOSSW06 scheme described in 7.3.2 is a secure unrestricted sequential aggregate signature scheme if the CDH assumption holds. This security is not proven in the original proof of the LOSSW06 scheme, and, as described above, the original proof of the LOSSW06 scheme does not imply the following proof. More precisely, whereas the proof goal of the original proof is to forge a signature in the Waters signature scheme, that of the following proof is to construct an algorithm to solve the CDH problem via a forgery output by an adversary $\mathcal{A}$. The following proof is our main contribution.

Theorem 28. The LOSSW06 scheme is $(t, q_c, q_s, \ell, n, \epsilon)$-secure if (t', ϵ')-CDH assumption holds in $\mathbb{G}$, where $\epsilon' = \frac{\epsilon}{16(\ell+1)q_s}$, $t' = t + t_{exp}((\ell+1)q_c + 6q_s + 2) + 2q_c t_P$, t_{exp} is a computational time for a single exponentiation and t_P is that for a single pairing computation.

Proof. Given a challenge $(g, g^a, g^b, p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$, $\mathcal{B}$ sets a key registration list $\mathcal{L} = \emptyset$ and $d = 4q_s$, where d affects a probability to solve the problem and we do not describe the detail here. Then, $\mathcal{B}$ generates $s \leftarrow \{0, \cdots, l\}$, ℓ-length vectors $(v_1^*, \cdots, v_\ell^*) \leftarrow \mathbb{Z}_d^\ell$, $(z_1^*, \cdots, z_\ell^*) \leftarrow \mathbb{Z}_p^\ell$ and $v' \leftarrow \mathbb{Z}_d, z' \leftarrow \mathbb{Z}_p$, and then sets up polynomials $G(m_i) = v' + \sum_{i=1}^{\ell} v_i^* m_{i,j} - ds$ and $K(m_i) = z' + \sum_{i=1}^{\ell} z_i^* m_{i,j}$ for these values, where $m_{i,j}$ is j-th bit in m_i. Next, $\mathcal{B}$ sets $(g_1 = g, g_2 = g^b)$ as public parameter, and publishes $V_{i^*,1} = (g^b)^{v_{i^*,1}} g^{z_{i^*,1}}, \cdots, V_{i^*,\ell} = (g^b)^{v_{i^*,\ell}} g^{z_{i^*,\ell}}$ as a public key pk^* of a target signer, where $\left(V_i' \prod_{j=1}^{\ell} V_{i^*,j}^{m_j}\right) = (g^b)^{G(m)} g^{K(m)}$ holds. $\mathcal{B}$ runs $\mathcal{A}$ with $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g_1, g_2, A^*, V^*, V_{i^*,1}, \cdots, V_{i^*,\ell})$ as input.

Certification Query: Given $sk_i = (g_2^{\alpha_i}, v_i', v_{i,1}, \cdots, v_{i,\ell})$, $pk_i = (A_i, V_i', V_{i,1}, \cdots, V_{i,\ell})$ by $\mathcal{A}$ for any signer, $\mathcal{B}$ check that $\boldsymbol{e}(g_2^{\alpha_i}, g_1) = \boldsymbol{e}(g_2, A_i), V_i' = g^{v_i'}, V_{i,1} = g^{v_{i,1}}, \cdots, V_{i,\ell} = g^{v_{i,\ell}}$ hold. If not $\mathcal{B}$ outputs $\perp$, otherwise registers $(A_i, V_i', V_{i,1}, \cdots, V_{i,\ell})$ in $\mathcal{L}$ as a public key pk_i.

Signing Query: In this proof, $\mathcal{B}$ checks only that that $\sum_{m_j \in \{m_{i^*}^{(h)}\}} G(m_j) \neq 0$ holds, where $\{m_{i^*}^{(h)}\}$ means a set of messages to be signed by pk^*. Here, we denote by Δ^* a set of indexes corresponding to pk^* in $\{pk_j\}_{j=1}^{i-1}$ and by $|\Delta^*|$ the size of the set.

If $\sum_{m_j \in \{m_{i^*}^{(h)}\}} G(m_j) \neq 0$ holds, $\mathcal{B}$ generates a random number $r \leftarrow \mathbb{Z}_p^*$ and executes the following computations:

$$
\begin{aligned}
R_i &= g^r (g^a)^{-\frac{|\Delta^*|}{\sum_{m_j \in \{m_{i^*}^{(h)}\}} G(m_j)}}, \\
S_i &= (g^a)^{-|\Delta^*| \frac{\sum_{m_j \in \{m_{i^*}^{(h)}\}} K(m_j)}{\sum_{m_j \in \{m_{i^*}^{(h)}\}} G(m_j)}} \left(\prod_{m_j \in \{m_{i^*}^{(h)}\}} V^* \prod_{e=1}^{\ell} V_{i^*,j}^{m_{j,e}} \right)^r \\
&\quad \times (R_i)^{\sum_{j=1 \wedge j \notin \Delta^*}^{i} \left(v_j' + \sum_{e=1}^{\ell} v_{j,e} m_{j,e}\right)} g_2^{\sum_{j=1 \wedge j \notin \Delta^*}^{i} \alpha_j}.
\end{aligned}
$$

This signature can be written as follows:

$$\begin{aligned} S_i &= (g^{ab})^{|\Delta^*|}\left((g^b)^{\sum_{m_j\in\{m_{i^*}^{(h)}\}}G(m_j)}g^{\sum_{m_j\in\{m_{i^*}^{(h)}\}}K(m_j)}\right)^{-\frac{|\Delta^*|a}{\sum_{m_j\in\{m_{i^*}^{(h)}\}}G(m_j)}} \\ &\quad\times\left(\prod_{m_j\in\{m_{i^*}^{(h)}\}}V^*\prod_{e=1}^{\ell}V_{i^*,j}^{m_{j,e}}\right)^r(R_i)^{\sum_{j=1\wedge j\notin\Delta^*}^{i}(v'_j+\sum_{e=1}^{\ell}v_{j,e}m_{j,e})}g_2^{\sum_{j=1\wedge j\notin\Delta^*}^{i}\alpha_j} \\ &= g_2^{\sum_{j=1}^{i}\alpha_j}\left(\prod_{j=1}^{i}\left(V'_j\prod_{e=1}^{\ell}V_{j,e}^{m_{j,e}}\right)\right)^{r-\frac{|\Delta^*|a}{\sum_{m_j\in\{m_{i^*}^{(h)}\}}G(m_j)}}. \end{aligned}$$

This signature (S_i, R_i) is accepted.

Output: After q_c iterations of the certification queries and q_s iterations of the signing queries, $\mathcal{A}$ outputs a forgery $(\{m_i^*\}_{i=1}^n, \sigma_n^*)$, where $\{m_{i^*}^*\}$ is a set of the messages to be signed by pk^*. $\mathcal{B}$ checks that $\sum_{m_j\in\{m_{i^*}^*\}}G(m_j)\neq 0$ holds for the forgery. If not, $\mathcal{B}$ aborts the process. Otherwise, $\mathcal{B}$ can solve the CDH problem as follows:

From the verification equation holds and the setup in **Initial Phase**, (S_n^*, R_n^*) can be written as follows:

$$\begin{aligned} R^* &= g^r, \\ S^* &= \left(g^{ab}\right)^{|\Delta^*|}\prod_{i=1\wedge i\notin\Delta^*}^{n}g^{\alpha_i}\cdot\left(g^{\sum_{m_j\in\{m_{i^*}^*\}}K(m_j)}\right)^r\left(\prod_{i=1\wedge i\notin\Delta^*}^{n}\left(V'_i\prod_{j=1}^{\ell}V_{i,j}^{m_{i,j}}\right)\right)^r, \end{aligned}$$

where r is a unknown random number. From the following computation, $\mathcal{B}$ can solves the CDH problem.

$$g^{ab}=\left(\frac{S^*}{\left(\prod_{j=1\wedge j\notin\Delta^*}^{n}g^{\alpha_j}\right)(R^*)^{\sum_{j=1\wedge j\notin\Delta^*}^{n}(v'_j+\sum_{e=1}^{\ell}v_{j,e})+\sum_{m_j\in\{m_{i^*}^*\}}K(m_j)}}\right)^{\frac{1}{|\Delta^*|}}.$$

Finally, we evaluate the success probability ϵ'. This analysis is almost the same as that in [95], and the probability is given as $\epsilon'\geq\epsilon\cdot\Pr[E_1]$, where

$$E_1 = \left[\bigwedge_{h=1}^{q_s}\left(\sum_{m_j\in\{m_{i^*}^{(h)}\}}G(m_j)\neq 0\right)\wedge\sum_{m_j\in\{m_{i^*}^*\}}G(m_j)=0\right].$$

Therefore, $\epsilon' \geq \epsilon \cdot \frac{1}{16(\ell+1)q_s}$ holds. The execution time of $\mathcal{B}$ is that of $\mathcal{A}$ plus $\ell+1$ exponentiation computations and two pairing computations for q_c certification queries, six exponentiation computations for q_s signing queries and two exponentiation computations in the final step. Therefore, $t' = t + t_{exp}((\ell+1)q_c + 6q_s + 2) + 2q_c t_P$ holds, where t_{exp} is a computational time of one exponentiation computation and t_P is a computational time of one pairing computation. □

Extension to Ordered Multisignatures

As described in Section 7.2, any aggregate signatures can be converted into ordered multisignatures by which each signer signs a concatenation of a message and public keys of a signing group. Hence, we can construct an unrestricted ordered multisignature scheme from the LOSSW06 scheme through the analysis in Section 7.3.2. The construction is provably secure without random oracles.

7.3.3 Discussion

Efficiency Comparison with Dividing-Out Method

Suppose the same signer generates two or more signatures successively in the unrestricted LOSSW06 scheme described in Section 7.3.2. Then the efficiency of such signature generation and aggregation can be improved by combining our new technique with the Lazy Verification technique. By utilizing the Lazy Verification technique, signer does not need to keep all the previous messages for signing. As one can observe in Section 7.3.2, a simpler procedure, the equation (7.1), can be used in the signing algorithm instead of the less efficient one, the equation (7.1). Intuitively, this is possible since the random number r_i used in the equation (7.1) does not need to be selected in every successive signing operation in the equation (7.1) and the same R_{i-1} can be treated as R_i. Moreover, the message m_{i-1} previously signed by the same signer does not appear in the formula for S_i in the equation (7.1) whereas the corresponding message appears in the last component of the formula for S_i in the equation (7.1). In the existing Dividing-out technique, the complicated procedure (7.1) is always required even for the case of successive generation of signatures by the same signer, and the messages previously signed by the same signer need to be kept by the signer during the signing operation. Hence the construction based on our technique can be significantly more efficient than the Dividing-out-based construction if

the same signer signs large amount of data consecutively as described in the practical applications of Section 7.1.3.

Reducing the Size of the Public Parameter and Public Key

In some applications, even the size of the public parameter and of public key creates a bottleneck. In order to reduce the size of these, we can adopt the approach of Naccache [70]. In this construction, the messages are divided into λ chunks, and the size of each chunk is 32 bits. For 128-bit security, the public parameter can be reduced from 256 generators to eight generators.

We omit the details of the scheme construction and the security proof, but these are similar to details in [70]. However, we note that the Naccache approach decreases the reduction cost in the security proof. Hence, we must utilize that approach carefully.

Batch Verification and Batch Identification

A batch verification algorithm [8] takes as input n signatures on n messages from n signers, and outputs *accept* if all individual signatures verify with probability 1 and *reject* otherwise with probability $1-2^{-\tau}$. Ferrara et al. [33] showed the batch verification of the LOSSW06 scheme and the BGLS03 scheme, and their batch verifications are useful for our analysis version. We recall the batch verification in [33].

Let a security parameter of the batch verification be τ, which in practice could be 80. It works as follows, where there are η signatures and we denote by $x^{(j)}$ j-th tuple about the signatures for all x:

$\mathbf{Batch}\left(\left(\{m_i^{(1)}\}_{i=1}^n, \sigma_n^{(1)}, \{pk_i^{(1)}\}_{i=1}^n\right), \cdots, \left(\{m_i^{(\eta)}\}_{i=1}^n, \sigma_n^{(\eta)}, \{pk_i^{(\eta)}\}_{i=1}^n\right)\right)$ For all $i = [1, n]$ and $j \in [1, \eta]$, parse $m^{(i)}$ as $(m_{j,1}^{(i)}, \cdots, m_{j,\ell}^{(i)})$ and $\sigma_n^{(i)}$ as $(S_n^{(i)}, R_n^{(i)})$. Then choose η random numbers $\delta_1, \cdots, \delta_\eta \in \{0,1\}^\tau$ and compute as follows:

$$\begin{aligned} \boldsymbol{e}\left(\prod_{j=1}^{\eta}(S_n^{(i)})^{\delta_j}, g_1\right) &\stackrel{?}{=} \prod_{j=1}^{\eta}\boldsymbol{e}\left(g_2, \prod_{i=1}^{n}(A_i^{(j)})^{\delta_j}\right) \cdot \\ &\quad \times \prod_{j=1}^{\eta}\boldsymbol{e}\left((R_n^{(j)})^{\delta_j}, \prod_{i=1}^{n}\left(V_j^{'(j)}\prod_{j=1}^{\ell}(V_{i,j}^{(j)})^{m_{i,j}}\right)\right). \end{aligned}$$

Theorem 29. The above algorithm is a batch verification for the LOSSW06 scheme with error $2^{-\tau}$.

As described in the next section, some applications such as the sensor networks require a large amount of the signatures, and the batch verifications are quite practical in this scenario. As a weakness of the batch verifications, if there is even a single invalid signature in the batch instance, the the batch algorithm will reject the entire batch with high probability. However, the weakness can be overcome by a divide-and-conquer approach [76]. In this approach, the verifier shuffles the incoming batch of the signatures, and if batch verifications fails, divide the collection into two halves. Then the verifier recurse on the halves. Hence, the invalid signatures can be detected.

7.4 Construction of Identity-Based Scheme

7.4.1 Overview

Based on our new proof idea explained in Section 7.3.1, it is expected that any sequential aggregate signature scheme (not only the LOSSW06 scheme [64]) secure in the restricted setting which satisfies the conditions (I) and (II) in Section 7.3.1 becomes secure in the unrestricted setting as well. A concrete example of such an existing scheme is the BGOY10 ID-based sequential aggregate signature scheme proposed [15], whose original security proof in the restricted setting was given in the random oracle model under the IBSAS-CDH assumption mentioned in Chapter 2. We note that random oracles can be interpreted as a special kind of programmable hash functions, therefore our new technique is also applicable to the schemes using random oracles. The reason why the original proof cannot be extended to the unrestricted setting, and how we resolve the problem by our new proof technique, are essentially the same as the case of the LOSSW06 scheme discussed in Section 7.3. To the authors' best knowledge, this is the first result to present an unrestricted ID-based sequential aggregate signature scheme (even in the random oracle model).

7.4.2 Syntax

We recall the syntax in [15]. Similarly as that in Section 7.2, we describe the definition in conjunction with the notion of lazy verification [18].

Setup(1^k) Given a security parameter 1^k, return a master secret key msk and its corresponding master public key mpk.

KeyDerivation(msk, mpk, ID_i) Given msk, mpk and an identity-string $ID_i \in \{0,1\}^*$, return a secret key sk_i for ID_i.

Signing$(mpk, sk_i, m_i, \boxed{\{m_j\}_{j=1}^{i-1}, \{ID_j\}_{j=1}^{i-1},} \sigma_{i-1})$ Given mpk, sk_i, a message $m_i \in \{0,1\}^*$ and an aggregate signature σ_{i-1} of $\{ID_j\}_{j=1}^{i-1}$ on $\{m_j\}_{j=1}^{i-1}$, return a new aggregate signature σ_i of $\{ID_j\}_{j=1}^{i}$ on $\{m_j\}_{j=1}^{i}$.

Verification$(\{ID_j, m_j\}_{j=1}^{i}, \sigma_i)$ Given i identity-message pairs and σ_i, output *accept* or *reject*.

Definition 30 (Correctness)**.** In an ID-based sequential aggregate signature scheme, we say that the scheme is correct if, for all msk, mpk given by **Setup** and sk_i given by **KeyGeneration** for all ID_i, **Verification**$(\{ID_j\}_{j=1}^{i}, \{m_j\}_{j=1}^{i}$, **Signing**$(mpk, sk_j, m_i, \{m_j\}_{j=1}^{i-1}, \{ID_j\}_{j=1}^{i-1}, \sigma_{i-1})$) outputs *accept* for all $i \in [1, n]$.

Lazy-verification construction is a construction removing the framed text in **Signing**. Similarly as Section 7.2, the correctness holds even for adopting lazy verification.

7.4.3 Security Model

We describe a security model of an unrestricted ID-based sequential aggregate signature scheme. Whereas the model described in [15] is for the restricted setting, we extend the model in [15] to the unrestricted one. Our model is a natural extension from the model in [15].

Initial Phase The challenger $\mathcal{C}$ generates a master secret key msk and its corresponding master public key mpk, and run $\mathcal{A}$ with mpk as input.

Key Derivation Query $\mathcal{A}$ send any string $ID_i^{(h)}$ to $\mathcal{C}$, and $\mathcal{C}$ return a secret key sk_i for $ID_i^{(h)}$.

Signing Query For all i, $\mathcal{A}$ generates a signing query $(mpk, m_i^{(h)}, ID_i^{(h)}, \sigma_{i-1})$ as h-th query, where the following conditions hold for the query: **Verification** algorithm outputs *accept*; $i - 1 < n$. Given such a query, $\mathcal{C}$ returns a signature σ_i.

Output After q_k iterations of **Key Derivation Query** and q_s iterations of **Signing Query**, $\mathcal{A}$ outputs a forgery $(\{(ID_i^*, m_i^*)\}_{i=1}^{n}, \sigma_n^*)$ where the following conditions hold: **Verification** algorithm outputs *accept*;

there exist at least ID^* such that $ID^* \notin \{ID_i^h\}_{h=1}^{q_k}$ holds; there exist a set of indexes Δ^* for (m^*, ID^*) such that $(m^*, ID^*) \notin \{(m_i^{(h)}, ID_i^h)\}_{h=1}^{q_s}$ holds. If all conditions hold, then $\mathcal{C}$ outputs *accept*. Otherwise, $\rfloor$ outputs $reject$.

Definition 31. We say that an ID-based sequential aggregate signature scheme is $(t, q_k, q_s, q_h, n, \epsilon)$-secure if there is no adversary $\mathcal{A}$ breaks with $(t, q_k, q_s, n, \epsilon)$. Here, we say that $\mathcal{A}$ breaks the scheme with $(t, q_k, q_s, n, \epsilon)$ as that a challenger $\mathcal{C}$ outputs *accept*, in the security game described above, with probability greater than ϵ with an execution time t. Here, $\mathcal{A}$ can generate at most q_k key derivation query, at most q_s signing queries and at most q_h random oracle queries, and n is the number of signers included in the forgery.

7.4.4 Unrestricted BGOY10 ID-based Sequential Aggregate Signature Scheme

In this section, we show the construction of the BGOY10 scheme [15]. The security of the scheme is guaranteed in the random oracle model, and the unrestricted setting of the scheme is our contribution.

Setup(1^k) Generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$. Choose random numbers $\alpha_1, \alpha_2 \in \mathbb{Z}_p$, cryptographic hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{G}$ and $H_3 : \{0,1\}^* \to \mathbb{Z}_p^*$, and compute $A_1 = g^{\alpha_1}$ and $A_2 = g^{\alpha_2}$. Output (α_1, α_2) as a master secret key msk and $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, A_1, A_2, H_1, H_2, H_3)$ as a master public key mpk.

Key Derivation(msk, mpk, ID_i) For a given string $ID_i \in \{0,1\}^*$, output $(H_1(ID_i)^{\alpha_1}, H_2(ID_i)^{\alpha_2})$ as a secret key sk_i.

Signing$(mpk, sk_i, m_i, \boxed{\{m_j\}_{j=1}^{i-1}, \{ID_j\}_{j=1}^{i-1},} \sigma_{i-1})$ Parse σ_{i-1} as $(S_{i-1}, R_{i-1}, W_{i-1})$. $\boxed{\text{Check if the verification algorithm return } accept \text{ for the query.}}$ $\boxed{\text{If not, output } \perp\text{. Otherwise,}}$ generate $r_i, x_i \in \mathbb{Z}_p$ and computes as follows:

$$\begin{aligned} W_i &= W_{i-1} \cdot g^{w_i}, \quad R_i = R_{i-1} \cdot g^{r_i}, \\ S_i &= S_{i-1} \cdot (W_{i-1})^{r_i} \cdot (R_i)^{w_i} \cdot H_1(ID_i)^{\alpha_1} \cdot H_2(ID_i)^{\alpha_2 \cdot H_3(ID_i \| m_i)}. \end{aligned}$$

Output (S_i, R_i, W_i).

Verification$(\{ID_j, m_j\}_{j=1}^{i}, \sigma_i)$ Parse σ_i as (S_i, R_i, W_i), and check that the following equation holds:

$$\boldsymbol{e}(S_i, g) \stackrel{?}{=} \boldsymbol{e}(R_i, W_i) \cdot \boldsymbol{e}\left(\prod_{j=1}^{i} H_1(ID_j), A_1\right) \cdot \boldsymbol{e}\left(\prod_{j=1}^{i} H_2(ID_j)^{H_3(ID_j \| m_j)}, A_2\right).$$

If so, output *accept*. Otherwise, output *reject*.

A Lazy verification in the above construction is to remove the framed text in **Signing**. In this construction, to generate a new aggregate signature, a set $\{m_j\}_{j=1}^{i-1}$ of messages and a set $\{ID_j\}_{j=1}^{i}$ of IDs are not required unless the verification algorithm is called for the signing process. Hence, the lazy verification approach is more practical.

7.4.5 Security Proof of the BGOY10 Scheme

Theorem 32. The BGOY10 scheme is $(t, q_k, q_s, q_{h_1}, q_{h_2}, q_{h_3}, n, \epsilon)$-secure if (t', q, ϵ')-IBSAS-CDH assumption holds, where $\epsilon' = \left(\epsilon - \frac{(q_s+q_{h_3})(q_s+q_{h_3}-1)}{2p}\right)\frac{1}{e(z+1)}$, $q = q_s$, $t' = t + t_{exp}(q_{h_1} + q_{h_2} + q_s(2(n+1)) + 2n + 2)$ and t_{exp} is a computational time for a single exponentiation.

Proof. The goal of this proof is to construct an algorithm $\mathcal{B}$ of solving the IBSAS-CDH problem. In this proof, we utilize Coron's technique [28] where an instance of the problem is randomly embedded with some distribution δ and δ is optimized at the end of the proof.

Given an instance $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, g^{a_1}, g^{b_1}, g^{a_2}, g^{b_2})$, $\mathcal{B}$ sets $A_1 = g^{a_1}$ and $A_2 = g^{a_2}$, and handles H_1, H_2 and H_3 as random oracles where H_1-List$[\cdot,\cdot,\cdot]$, H_2-List$[\cdot,\cdot,\cdot]$, and H_3-List$[\cdot,\cdot]$ are hash tables. $\mathcal{B}$ runs $\mathcal{A}$ with $mpk = (p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, A_1, A_2, H_1, H_2, H_3)$ as input and interacts as follows:

H_1**-Hash Query,** H_2**-Hash Query**: Given any string $ID_i \in \{0,1\}^*$, check that ID_i has been already queried. If so, return a corresponding value from H_1-List and H_2-List. Otherwise, pick a random bit $b_i \leftarrow \{0,1\}$ with some probability δ which assigns $b_i = 1$, and generate $(\gamma_{i,1}, \gamma_{i,2}) \leftarrow \mathbb{Z}_p^2$. If $b_i = 1$, then set $H_1[ID_i] = g^{\gamma_{i,1}}$ and $H_2[ID_i] = g^{\gamma_{i,2}}$. Otherwise, set $H_1[ID_i] = g^{b_1} g^{\gamma_{i,1}}$ and $H_2[ID_i] = g^{b_2} g^{\gamma_{i,2}}$. Register $(ID_i, b_i, \gamma_{i,1})$ in H_1-List and $(ID_i, b_i, \gamma_{i,2})$ in H_2-List, and return $H_1[ID_i]$ and $H_2[ID_i]$.

H_3**-Hash Query**: Given any string $ID_i \| m_i$, generate a random number $d_i \leftarrow \mathbb{Z}_p^*$ and register $(ID_j \| m_i, d_i)$ in H_3-List.

Key Derivation Query: Given any string $ID_i \in \{0,1\}^*$, check b_i for ID_i in H_1-List. If $b_i = 0$, then abort the process. Otherwise, return $H_1(ID_i)^{a_1} = (g^{a_1})^{\gamma_{i,1}}$ and $H_2(ID_i)^{a_2} = (g^{a_2})^{\gamma_{i,2}}$.

Signing Query: Given $(ID_i, m_i, \{(ID_j, m_j)\}_{j=1}^{i-1}, \sigma_{i-1})$, check b_i for ID_i in H_1-List. If $b_i = 1$, then retrieve $\gamma_{i,1}, \gamma_{i,2}$ and d_i from each hash table, and generate $r, w \leftarrow \mathbb{Z}_p$ and compute as follows:

$$\begin{aligned} R_i &= R_{i-1} \cdot g^r, \\ W_i &= W_{i-1} \cdot g^w, \\ S_i &= S_{i-1} \cdot (W_{i-1})^r \cdot (R_i)^w (g^{a_1})^{\gamma_{i,1}} (g^{a_2})^{\gamma_{i,2} d}. \end{aligned}$$

Otherwise, check if there exists ID_j in $\{(ID_j, m_j)\}_{j=1}^{i-1}$ such that $ID_i \neq ID_j \wedge b_j = 0$ for $1 \leq j < i$. If so, abort the process. Otherwise, let Δ_i be a set of indexes corresponding to ID_i in $\{(ID_j, m_j)\}_{j=1}^{i}$ and compute as follows:

$$\begin{aligned} (S', R', W') &\leftarrow \mathcal{O}^{IBSAS-CDH}_{g, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}} \left(\frac{\sum_{j \in \Delta_i} d_j}{|\Delta_i|} \right) \\ R_i &= (R')^{|\Delta_i|}, \quad W_i = W', \\ S_i &= (S')^{|\Delta_i|} \prod_{j=1}^{i} (g^{a_1})^{\gamma_{j,1}} (g^{a_2})^{\gamma_{j,2} d_j}. \end{aligned}$$

These values can be written as follows and hence are a valid signature.

$$\begin{aligned} S_i &= \left(g^{rx} g^{a_1 b_1} g^{\frac{\sum_{j \in \Delta_i} d_j}{|\Delta_i|} a_2 b_2} \right)^{|\Delta_i|} \prod_{j=1}^{i} (g^{a_1})^{\gamma_{j,1}} (g^{a_2})^{\gamma_{j,2} d_j} \\ &= g^{rx|\Delta_i|} (g^{b_1})^{a_1 |\Delta_i|} (g^{b_2})^{a_2 \sum_{j \in \Delta_i} d_j} \left(\prod_{j \in \Delta_i} (g^{a_1})^{\gamma_{j,1}} (g^{a_2})^{\gamma_{j,2} d_j} \right) \\ &\quad \times \left(\prod_{j=1 \wedge j \notin \Delta_i}^{i} H_1(ID_j)^{a_1} H_2(ID_j)^{a_2 H_3[ID_i \| m_i]} \right) \\ &= g^{rx|\Delta_i|} \left(\prod_{j \in \Delta_i} \left(g^{b_1} g^{\gamma_{j,1}} \right)^{a_1} \left(g^{b_2} g^{\gamma_{j,2}} \right)^{a_2 d_j} \right) \\ &\quad \times \left(\prod_{j=1 \wedge j \notin \Delta_i}^{i} H_1(ID_j)^{a_1} H_2(ID_j)^{a_2 H_3[ID_i \| m_i]} \right) \\ &= g^{rx|\Delta_i|} \left(\prod_{j=1}^{i} H_1(ID_j)^{a_1} H_2(ID_j)^{a_2 H_3[ID_i \| m_i]} \right) \end{aligned}$$

Output: Given a forgery $(\{(ID_i^*, m_i^*)\}_{i=1}^n, \sigma_n^*)$ by $\mathcal{A}$, check that there exists a set Δ^*, where $|\Delta^*| > 0$, of indexes corresponding to ID_i^* such that $b_i = 0$. If not, abort the process. Otherwise, compute as follows:

$$\begin{aligned}
S' &= \frac{S^*}{\prod_{i=1}^n (g^{a_1})^{\gamma_{i,1}} (g^{a_2})^{\gamma_{i,2} H_3[ID_i \| m_i]}} = g^{r^* w^*} \prod_{j \in \Delta^*} \left(g^{b_1}\right)^{a_1} \left(g^{b_2}\right)^{a_2 H_3(ID_j \| m_j)} \\
&= g^{r^* w^*} g^{a_1 b_1 |\Delta^*|} g^{a_2 b_2 \sum_{j \in \Delta^*} H_3(ID_j \| m_j)} \\
\therefore S &= \left(S'\right)^{\frac{1}{|\Delta^*|}} = g^{\frac{r^*}{|\Delta^*|} w^*} g^{a_1 b_1} g^{a_2 b_2 \frac{\sum_{j \in \Delta^*} H_3(ID_j \| m_j)}{|\Delta^*|}}, \\
R &= (R^*)^{\frac{1}{|\Delta^*|}} = g^{\frac{r^*}{|\Delta^*|}}, \\
W &= W^*.
\end{aligned}$$

Output $\left(\frac{\sum_{j \in \Delta^*} H_3(ID_j \| m_j)}{|\Delta^*|}, S, R, W\right)$ as a solution of the IBSAS-CDH problem.

The success probability ϵ' and the execution time t' of $\mathcal{B}$ can be obtained similarly as that of the proof in [15]. Here, let *Collide* be an event that $\mathcal{B}$ outputs (m, S, R, Z) such that it has been queried to $\mathcal{O}^{IBSAS-CDH}_{g, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}}$, let $forge$ be an event that $\mathcal{A}$ outputs a forgery, and let *abort* be an event that $\mathcal{B}$ aborts. Then, the following equation can be obtained as follows:

$$\begin{aligned}
\epsilon' &\geq \Pr[forge \wedge \overline{collide} \wedge \overline{abort}] \\
&= \Pr[forge \wedge \overline{collide} | \overline{abort}] \cdot \Pr[\overline{abort}] \\
&= \Pr[forge \backslash collide | \overline{abort}] \cdot \Pr[\overline{abort}] \\
&= \left(\Pr[forge | \overline{abort}] - \Pr[collide | \overline{abort}]\right) \cdot \Pr[\overline{abort}]
\end{aligned}$$

From the definition of the adversary $\mathcal{A}$, $\Pr[forge|\overline{abort}] - \Pr[collide|\overline{abort}]$ is ϵ. $\Pr[collide|\overline{abort}]$ can be obtained from the birthday paradox, and thus $\Pr[collide|\overline{abort}] = \frac{(q_s+q_{h_3})(q_s+q_{h_3}-1)}{2p}$ holds. The steps that $\mathcal{B}$ aborts are in **Key Derivation, Query**, **Signing Query** and **Output**. In particular, while the aborts in **Key Derivation Query** and **Signing Query** can be avoided as long as $b_i = 1$ holds, there must exist ID_i in **Output** such that $b_i = 0$ holds. Hence, $\Pr[\overline{abort}] = \delta^{q_k} \cdot \delta^{q_s n}(1-\delta)$ holds. Here, we define a function $f(\delta) := \delta^{q_k} \cdot \delta^{q_s n}(1-\delta)$. From its derived function, $f(\delta)$ is optimized at $\delta_{opt} = \frac{q_k+q_s n}{q_k+q_s n+1}$, and thus the following equation holds from the definition of the base of natural logarithm where we denote by $z := q_k + q_s n$ for short:

$$\begin{aligned}
f(\delta_{opt}) &= \left(\frac{z}{z+1}\right)^z \left(1 - \frac{z}{z+1}\right) \geq \frac{1}{e(z+1)} \quad \because \lim_{z \to \infty} \left(\frac{z}{z+1}\right)^z = \frac{1}{e}, \\
\therefore \epsilon' &\geq \left(\epsilon - \frac{(q_s+q_{h_3})(q_s+q_{h_3}-1)}{2p}\right) \frac{1}{e(z+1)}.
\end{aligned}$$

The number of queries to the oracle $\mathcal{O}^{IBSAS-CDH}_{g,g^{a_1},g^{a_2},g^{b_1},g^{b_2}}$ is at most q_s signing queries, and thus $q = q_s$ holds. The computational time of $\mathcal{B}$ is that of $\mathcal{A}$ plus exponentiations in H_1-**Hash Query**, H_2-**Hash Query**, **Signing Query** and **Output**, and thus the following equation can be obtained:

$$t' = t + t_{exp}(q_{h_1} + q_{h_2} + q_s(2(n+1)) + 2n + 2),$$

where t_{exp} is the computational time of one exponentiation. □

7.5 Unrestricted Ordered Multisignature Scheme

7.5.1 Syntax

We recall the system model in [14]. Ordered multisignature is a digital signature where each a member among a signing group sings a common message and its position in the group. Ordered multisignature scheme consists of the following algorithms.

Setup(1^k) Given a security parameter 1^k, return a public parameter $para$.

KeyGeneration$(para)$ Given $para$, return a secret key sk_i and its corresponding public key pk_i.

Signing$(sk_i, m, \psi_{i-1}, \sigma_{i-1}, pk_i)$ Given a secret key sk_i, a message m, a signing order ψ_{i-1} from the first signer to ith signer, an ordered multisignature σ_{i-1} on m in ψ_{i-1} and a public key pk_i, return a signature σ on m in ψ_i.

Verification$(m, \psi_i, \sigma_i, \{pk_j\}_{j=1}^{i})$ Given m, ψ_i σ_i and a set $\{pk_j\}_{j=1}^{i}$ of public keys, return *accept* or *reject*.

Definition 33 (Correctness)**.** In an ordered multisignature scheme, we say that the scheme is correct if, for all $para, sk_i$ and pk_i given by **Setup** and **KeyGeneration**, **Verification** $((m, \psi_i,$ **Signing** $(sk_i, m, \psi_{i-1}, \sigma_{i-1}, pk_i),$ $\{pk_j\}_{j=1}^{i})$ outputs *accept* for all $i \in [1, n]$.

7.5.2 Security Model for Unrestricted Ordered Multisignature Scheme

The following security model is defined in [14]. An ordered multisignature scheme must guarantee unforgeability with respect to the order of signers

in addition to that of messages. Namely, it should not be possible to reorder the positions of honest signers in an ordered multisignature scheme, even if all other signers are malicious. Whereas the existing scheme is provably secure in the restricted setting, the security of unrestricted ordered multisignature scheme can be guaranteed via the following model.

There exist an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ in this model. The challenger $\mathcal{C}$ has a certified-key list $\mathcal{L}$ to register users and their own public keys, including and $\mathcal{A}$ can get to know all the keys in $\mathcal{L}$ except for the one given by $\mathcal{C}$ to the target signer. The advantage of $\mathcal{A}$ can be obtained with the probability that $\mathcal{C}$ outputs *accept* in the subsequent game. Hereinafter, we denote by $x^{(i)}$ the value of the i-th query for all x.

Initial Phase The challenger $\mathcal{C}$ generates a public parameter $para$ by **Setup** and a pair of challenge key (sk^*, pk^*) of a target signer by **KeyGeneration**. Then, $\mathcal{C}$ initializes $\mathcal{L} := \emptyset$, and runs $\mathcal{A}$ with $para$ and pk^* as input.

Certification Query $\mathcal{A}$ generates sk_i and its corresponding public key pk_i for any signer. Then, $\mathcal{A}$ provides (sk_i, pk_i) to $\mathcal{C}$, and $\mathcal{C}$ registers pk_i in $\mathcal{L}$.

Signing Query For all i, $\mathcal{A}$ generates a signing query $(m^{(h)}, \psi_{i-1}^{(h)}, \sigma_{i-1}, pk^*)$ as h-th query for the target signer, where the following conditions hold for the query and for all h: **Verification** algorithm outputs *accept*; For all pk_j, pk_j in $\psi_{i-1}^{(h)}$ is included in $\mathcal{L}$; $i-1 < n$. Given $(m^{(h)}, \psi_{i-1}^{(h)}, \sigma_{i-1}, pk^*)$ by $\mathcal{A}$, $\mathcal{C}$ runs **Signing** $(sk_i, m^{(h)}, \psi_{i-1}^{(h)}, \sigma_{i-1}, pk_i)$, and obtains σ_i and $\psi_i^{(h)} = \psi_{i-1}^{(h)} \parallel pk^*$. Finally, $\mathcal{C}$ returns σ_i on $m^{(h)}$ in $\psi_i^{(h)}$.

Output After q_c iterations of the certification queries and q_s iterations of the signing queries, $\mathcal{A}$ outputs a forgery $(m^*, \psi_n^*, \sigma_n^*)$. Here, let the target signer be i^*th signer in ψ_n^*, and the following conditions hold for the forgery: **Verification**$(m^*, \psi_n^*, \sigma^*, \{pk_i\}_{i=1}^n)$ outputs *accept*; there exist at least one index $j \in \Delta^*$ such that $(m^*, \psi_j^*) \notin \{(m^{(h)}, \psi_i^{(h)})\}_{h=1}^{q_s}$ holds, where Δ^* is a set of indexes corresponding to pk^* in ψ_n^* and each ψ_j^* is extracted from ψ_n^* as a signer structure from the first signer to the target signer; ψ_n^* includes pk^*; For all pk_j, pk_j in ψ_n^* is included in $\mathcal{L}$. If all conditions hold, then $\mathcal{C}$ outputs *accept*. Otherwise, $\mathcal{C}$ outputs *reject*.

Definition 34. We say that an ordered multisignature scheme is $(t, q_c, q_s, \ell, n, \epsilon)$-secure if there is no adversary $\mathcal{A}$ breaks with $(t, q_c, q_s, \ell, n, \epsilon)$. Here, we define that $\mathcal{A}$ breaks the scheme with $(t, q_c, q_s, \ell, n, \epsilon)$ as that a challenger $\mathcal{C}$ outputs *accept*, in the security game described above, with the probability greater than ϵ within the execution time t. Here, $\mathcal{A}$ can generate at most q_c certification queries and at most q_s signing queries, ℓ is the length of the message output by $\mathcal{A}$, and n is the number of signers included in the forgery.

7.5.3 Insecurity of the BGOY07 Scheme

Review of the BGOY07 Ordered Multisignature Scheme

We briefly review the BGOY07 scheme [14].

Setup(1^k) Generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$. Then, choose a random generator $g \in \mathbb{G}$ and a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}$. Output $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, H)$ as the public parameter *para*.

KeyGeneration$(para)$ Pick random numbers $\alpha_i, t_i, v_i \leftarrow \mathbb{Z}_p$ and compute $A_i = g^{\alpha_i}$, $T_i = g^{t_i}$, $V_i = g^{v_i}$. sk_i is (α_i, t_i, v_i) and its corresponding public key pk_i is (A_i, T_i, V_i).

Signing$(sk_i, m, \psi_{i-1}, \sigma_{i-1}, pk_i)$ Parse the secret key sk_i as (α_i, t_i, v_i), ψ_{i-1} as $\{pk_j\}_{j=1}^{i-1}$ and the previous signature σ' as (S_{i-1}, R_{i-1}), where pk_j as (A_i, T_i, V_i) for $j = [1, i]$. Pick a random number $r \leftarrow \mathbb{Z}_p$ and compute as $R_i \leftarrow R_{i-1} \cdot g^r$ and $S_i \leftarrow S_{i-1} \cdot H(m)^{\alpha_i}(R_i)^{t_i + iv_i}\left(\prod_{j=1}^{i-1} T_j(V_j)^j\right)^r$. Set $\psi_i = \psi_{i-1} \parallel pk_i$ and output $\sigma = (S_i, R_i)$ as a signature on m in ψ_i.

Verification$(m, \psi_i, \sigma_i, \{pk_j\}_{j=1}^{i})$ Parse σ_i as (S_i, R_i) and pk_i as (A_j, T_j, V_j) for $j = [1, i]$. Check that $\boxed{\text{all of } \{pk_j\}_{j=1}^{i} \text{ are distinct and output } reject}$ $\boxed{\text{if not. Otherwise, check that}}$ the following equation holds:

$$\boldsymbol{e}(S_i, g) \stackrel{?}{=} \boldsymbol{e}\left(H(m), \prod_{j=1}^{i} A_j\right) \cdot \boldsymbol{e}\left(\prod_{j=1}^{i} T_j(V_j)^j, R_i\right).$$

If the previous equation holds, output *accept*. Otherwise, output *reject*.

The unrestricted setting is a construction removing the framed text z in **Verification**. Similarly as the LOSSW06 scheme, removing the check of the public keys is out of the scope of the original construction of the BGOY07 scheme.

Insecurity under Unrestricted Setting

The signature equation in the BGOY07 scheme can be written as $S = H(m)^{\sum_{j=1}^{i} \alpha_i} \left(\prod_{j=1}^{i} T_j^j V_j\right)^r$ for any i. If signers are allowed to sign in multiple positions for one signature generation, then boss‖subordinate‖boss is allowed as the signing order, for example. Such a multiple appearance of signers encourages the following attack described below. In the case of ψ_3 =boss‖subordinate‖boss, the signature equation for ψ_3 becomes $S = H(m)^{2\alpha_a+\alpha_b} \left(T_a^4 V_a^2 T_b^2 V_b\right)^r$, where α_a is the secret key of the boss and α_b is that of the subordinate. Then, a malicious signer can forge the signature of the subordinate for another signing order ψ_2 =subordinate‖boss as follows: the attacker computes $S^{\frac{1}{2}} H(m)^{\frac{\alpha_b}{2}} (R)^{\frac{v_b}{2}}$, where v_b is an individual secret key corresponding to V_b, and this value is equal to $H(m)^{\alpha_a+\alpha_b} \times \left(T_b^1 V_b T_a^2 V_a\right)^r$, which is accepted on m in ψ_2. As noted in Section 7.1.2, the proof by Boldyreva et al. [14] is correct but this attack is outside the proof.

Chapter 8

Conclusion

In this paper, we discussed constructions of signature schemes for multiple signers. Signature scheme for multiple signers has been expected to provide various applications as a generic construction of digital signature. In this work, we discuss these constructions from the viewpoint of (1) optimized constructions for the applications and (2) mathematical property toward a generic construction. In particular, we considered, for the former case, an ordered multisignature scheme, a structured multisignature scheme, a BGP-aiding aggregate signature scheme and a certificateless aggregate signature scheme, and for the latter case an unrestricted aggregate signature scheme. As the results, we pointed out the problems of the existing schemes and solved the open problem described in each construction. More precisely, we proposed the ordered multisignature scheme without random oracles in Chapter 3, the structured multisignature scheme without a restriction in the number of signers in Chapter 4, the BGP-aiding aggregate signature scheme optimized for the specification of BGP in Chapter 5, and the certificateless aggregate signature scheme secure against the super adversaries who are the strongest adversary in Chapter 6. Meanwhile, in Chapter 7, we also discussed a construction which allows each signer to sign multiple times. The construction consists of common properties of the schemes from Chapter 3 to Chapter 6, and the properties seem to be a generic property to prove the security of the schemes for multiple signers. The properties are programmable hash function [40] and re-randomization, and are implied in the random oracle model. Finally, we describe an open problem. The problem is to construct a generic construction of signature scheme for multiple signers by utilizing the formalization of the re-randomization described in [16]. We consider such a construction is a properly generic construction

of digital signature.

Bibliography

[1] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. Synchronized aggregate signatures: New definitions, constructions and applications. In *Proc. of the 17th ACM Conference on Computer and Communications Security*, pages 473–484. ACM, October 2010. Full paper is available in Cryptology ePrint Archive, `http://eprint.iacr.org/2010/422`.

[2] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In *Proc. of the 9th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2003), Taipei, Taiwan*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Spirnger-Verlag, November-December 2003.

[3] APNIC. `http://www.apnic.net/`.

[4] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring multipath routing in the internet. *IEEE/ACM Transactions on Networking*, 19(3):830–840, June 2011.

[5] Ali Bagherzandi and Stanislaw Jarecki. Multisignatures using proofs of secret key possession, as secure as the diffie-hellman problem. In *Proc. of the 6th International Conference on Security and Cryptography for Networks (SCN 2008), Amalfi, Italy*, volume 5229 of *Lecture Notes on Computer Science*. Springer-Verlag, September 2008.

[6] Ali Bagherzandi and Stanislaw Jarecki. Identity-based aggregate and multisignature schemes based on rsa. In *Proc. of the 13th International Conference on Practice and Theory in Public Key Cryptography (PKC 2010), Paris, France*, volume 6056 of *Lecture Notes in Computer Science*, pages 480–498. Springer-Verlag, May 2010.

[7] Josep Balasch, Alfredo Rial, Carmela Troncoso, Bart Preneel, Ingrid Verbauwhede, and Christophe Geuens. Pretp: Privacy-preserving electronic toll pricing. In *Proc. of the 19th Unenix Security Symposium (Usenix Security 2010), Washington, USA*, volume 10. Usenix, August 2010.

[8] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Proc. of the first International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT 1998), Espoo, Finland*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer-Verlag, May 1998.

[9] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggrgate signatures. In *Proc. of the 34th International Colloquium on Automata, Languages and Programming (ICALP 2007), Wroclaw, Poland*, volume 4596 of *Lecture Notes in Computer Science*, pages 411–422. Springer-Verlag, July 2007.

[10] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, USA*, pages 390–399. ACM, October - November 2006.

[11] Mihir Bellare and Phillip Rogaway. Random oracle are practical: A paradigm for designing efficient protocols. In *Proc. of the 1st ACM Conference on Computer and Communications Security (CCS 1993), Fairfax, USA*, pages 62–73. ACM, November 1993.

[12] Hans L. Bodlaender and Babette de Fluiter. Parallel algorithms for series parallel graphs. In *Proc. of the Fourth Annual European Symposium (ESA 1996), Barcelona, Spain*, volume 1136 of *Lecture Notes in Computer Science*, pages 277–289. Springer-Verlag, September 1996.

[13] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Proc. of the 6th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2003), Miami, USA*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, January 2003.

[14] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing (extended abstract). In *Proc. of the 14th ACM Conference on Computer and Communication Security (CCS 2007), Alexandria, USA*, pages 276–285. ACM, October-November 2007. Full paper is available in `http://www.cc.gatech.edu/~aboldyre/papers/bgoy.pdf`.

[15] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing (extended abstract), 2010. (full paper).

[16] Alexandra Boldyreva and Robert Lychev. Provable security of s-bgp and other path vector protocols: model, analysis and extensions. In *Proc. of the 19th ACM Conference on Computer and Communications Security (CCS 2012), Borth Carolina, USA*, pages 541–552. ACM, October 2012. Full paper is available in Cryptology ePrint Archive.

[17] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. of the 22th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2003), Warsaw, Poland*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, May 2003.

[18] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations. In *Proc. of the 18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2012), Beijing, China*, volume 7658 of *Lecture Notes in Computer Science*, pages 663–680, December 2012. Full paper is available in Cryptology ePrint Archieve, `http://eprint.iacr.org/2011/222`.

[19] Mike Burmester, Yvo Desmedt, Hiroshi Doi, Masahiro Mambo, Eiji Okamoto, Mitsuru Tada, and Yuko Yoshifuji. A structured elgamal-type multisignature scheme. In *Proc. of the 9th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2000), Melbourne, Australia*, volume 1751 of *Lecture Notes in Computer Science*, pages 466–483. Springer-Verlag, January 2000.

[20] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, July 2004.

[21] Rafael Castro and Ricardo Dahab. Efficient certificateless signatures suitable for aggregation, December 2007. Cryptology ePrint Archive: Listing for 2007.

[22] Daniele Catteddu and Giles Hogben. Cloud computing risk assessment. Technical report, European Uion Agency for Network and Information Security, November 2009.

[23] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In *Proc. of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC 2003), Miami, USA*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, January 2003.

[24] Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *Proc. of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2006), St. Petersburg, Russia*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, May 2006.

[25] Ying-Ju Chi, Ricardo Oliveira, and Lixia Zhang. Cyclops: The as-level connectivity observatory. *ACM Sigcomm Computer Communication Review*, 38(4):5–16, October 2008.

[26] Eikoh Chida, Takao Nishizeki, Motoji Ohmori, and Hiroki Shizuya. On the one-way algebraic homomorphism. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E79-A(1):54–60, January 1996.

[27] Jean-Sébastien Coron. On the exact security of full domain hash. In *Proc. of the 20th Annual International Cryptology Conference (CRYPTO 2000), California, USA*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, August 2000.

[28] Jean-Sébastien Coron. On the exact security of full domain hash. In *Proc. of the 20th Annual International Cryptology Conference (CRYPTO 2000), California, USA*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-verlag, August 2000.

[29] Antonio de la Oliva, Marcelo Bagnulo, Alberto Garcia-Martinez, and Ignacio Soto. Performance analysis of the reachability protocol for ipv6 multihoming. In *Proc. of NEW2AN 2007*, volume 4712 of *LNCS*, pages 443–454. Springer, 2007.

[30] Hiroshi Doi, Masahiro Mambo, and Eiji Okamoto. Multisignature schemes using structured group id. In *Technical Report of IEICE*, volume 98, pages 43–48. IEICE, December 1998.

[31] Bennian Dou, Hong Zhang, Chungen Xu, and Mu Han. Identity-based sequential aggregate signature from rsa. In *Proc. of the 4th ChinaGrid Annual Conference (ChinaGrid 2009), Yantai, China*, pages 123–127. IEEE, August 2009.

[32] Nick Feamster, Hari Balakrishnan, and Jennifer Rexford. Some foundational problems in interdomain routing. In *Proc. of HotNets-3 2004*. ACM, November 2004. Available in `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.132.3571&rep=rep1&type=pdf`.

[33] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen. Practical short signature batch verification. In *Proc. of the 9th Cryptographers ' Track at the RSA Conference (CT-RSA 2009), San Francisco, USA*, volume 5473 of *Lecture Notes in Computer Science*, pages 309–324. Springer-Verlag, April 2009.

[34] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Proc. of the 25th Annual International Cryptology Conference (CRYPTO 2005), California, USA,*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer-Verlag, August 2005.

[35] Marc Fischlin, Anja Lehmann, and Dominique Schrōder. History-free sequential aggregate signatures. In *Proc. of the 8th Internation Conference on Security and Cryptography for Networks (SCN 2012), Amalfi, Italy*, volume 7485 of *Lecture Notes in Computer Science*, pages 113–130. Springer-Verlag, September 2012. Full paper is available in Cryptology ePrint Archieve, `http://eprint.iacr.org/2011/231`.

[36] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Proc. of the 9th International Conference on Theory and*

Practice of Public-Key Cryptography (PKC 2006) New York, USA, volume 3958 of *Lecture Notes in Computer Science*, pages 257–273. Springer-Verlag, April 2006.

[37] Michael Gerbush, Allison Lewko, Adam O'Neill, and Brent Waters. Dual form signatures: An approach for proving security from static assumptions. In *Proc. of the 18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2012), Beijing, China*, volume 7658 of *Lecture Notes in Computer Science*, pages 25–42. Springer-Verlag, December 2012. Full paper is available in Cryptology ePrint Archive, `http://eprint.iacr.org/2012/261`.

[38] Marc Girault. Self-certified public keys. In *Proc. of the 10th Theory and Applications of Cryptographic Techniques (EUROCRYPT 1991), Brighton, UK*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Spriger-Verlag, April 1991.

[39] Zheng Gong, Yu Long, Xuan Hong, and Kefei Ghen. Practical certificateless aggregate signatures from bilinear maps. *Journal of Information Science and Engineering*, 26(6):2093–2106, 2010.

[40] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In *Proc. of the 28th Annual Internatinal Cryptology Conference (CRYPTO 2008), Santa Barbara, USA*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer-Verlag, August 2008.

[41] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In *Proc. of the 33rd Annual Cryptology Conference (CRYPTO 2013), Santa Barbara, CA, USA*, volume 8042 of *Lecture Notes in Computer Science*, pages 494–512. Springer-Verlag, August 2013. Full paper is available in `http://eprint.iacr.org/2013/434`.

[42] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures u nder standard assumptions. In *Proc. of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2009), Cologne, Germany*, volume 5479 of *Lecture Notes in Computer Science*, pages 333–350. Springer-Verlag, April 2009. Full paper is available in `http://eprint.iacr.org/2009/028`.

[43] Wei hua Hou. An ordered multisignature without random oracles. In *Proc. of 2010 International Conference on Communications and Mobile Computing (CMC 2010), Shenzhen, China*, volume 1, pages 21–25. IEEE, April 2010.

[44] Xinyi Huang, Yi Mu, Willy Susilo, Duncan Wong, and Wei Wu. Certificateless signature revisited. In *Proc. of the 12th Australasian Conference on Information Security and Privacy (ACISP 2007), Townsville, Australia,*, volume 4586 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, July 2007.

[45] Jung Yeon Hwang, Dong Hoon Lee, and Moti Yung. Universal forgery of the identity-based sequential aggregate signature scheme. In *Proc. of the 4th ACM Conference on Computer and Communications Security (ASIACCS 2009), Sydney, Australia*, pages 157–160. ACM, March 2009.

[46] IANA. Autonomous system (as) numbers, 2013. `http://www.iana.org/assignments/as-numbers/as-numbers.xhtml`.

[47] Kenta Ishii, Kazutaka Saito, Akira Kanaoka, Naoki Kanayama, and Eiji Okamoto. General purpose c library for pairing cryptography. In *Proc. of the 30th Symposium on Cryptography and Information Security (SCIS 2013), Kyoto, Japan*. IEICE, January 2013. (Japanese Only).

[48] Kazuharu Itakura and Katsuhiro Nakamura. A public-key cryptosystem suitable for digital multi-signatures. *NEC Research and Development*, 71:1–8, 1983.

[49] Akira Kanaoka, Masayuki Okada, Yasuharu Katsuno, and Eiji Okamoto. Probabilistic packet marking method considering topology property for efficiency re-building dos attack paths. *TIPSJ*, 52(3):929–939, 2011.

[50] Bo Gyeong Kang, Je Hong Park, and Sang Geun Hahn. A certificate-based signature scheme. In *Proc. of the Cryptographers' Track at the RSA Conference (CT-RSA 2004), San Francisco, USA*, volume 2964 of *Lecture Notes in Computer Science*, pages 99–111. Springer-Verlag, February 2004.

[51] Kei Kawauchi and Mitsuru Tada. On the security and the efficiency of multi-signature schemes based on a trapdoor one-way permutation.

IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E88-A(5):1274–1282, May 2005.

[52] Stephen Kent, Charles Lynn, and Karen Seo. Secure border gateway protocol. *IEEE Journal of Selected Areas in Communications*, 18(4):582–592, 2000.

[53] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Proc. of the 4th Network and Distributed System Security Symposium (NDSS 2000), California, USA*. The Internet Society, February 2000. `http://www.isoc.org/isoc/conferences/ndss/2000/proceedings/042.pdf`.

[54] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. Aggregating cl signatures revisited: Extended functionality and better efficiency. In *The 17th International Conference on Financial Cryptography and Data Security (FC 2013), Okinawa, Japan*. Springer-Verlag, March - April 2013. Available in `http://fc13.ifca.ai/proc/5-2.pdf`.

[55] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. Sequential aggregate signatures with short public keys: Design, analysis and implementation studies. In *The 16th International Conference on Practice and Theory in Public-Key Cryptography (PKC 2013), Nara, Japan*, volume 7778 of *Lecture Notes in Computer Science*, pages 423–442. Springer-Verlag, February - March 2013.

[56] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was rong, whit is right. Cryptology ePrint Archive: Listing for 2012, February 2012. http://eprint.iacr.org/2012/064.

[57] Matt Lepinski. An infrastructure to support secure internet routing, 2012. RFC 6480.

[58] Matt Lepinski. Bgpsec protocol specification, 2013. Internet Draft, `http://datatracker.ietf.org/doc/draft-ietf-sidr-bgpsec-protocol/`.

[59] Matt Lepinski and Sean Turner. An overview of bgpsec, 2011. Internet Draft, `http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-overview-01`.

[60] Jiguo Li, Xinyi Huang, Yi Mu, Willy Susilo, and Qianhong Wu. Certificate-based signature: Security model and efficient construction. In *Proc. of the 4th European Workshop on Public Key Infrastructure (EuroPKI 2007), Palma de Mallorca*, volume 4582, pages 110–125. Springer-Verlag, June 2007.

[61] Xiangxue Li, Longjun Zhang, and Shiqun Li. Proxy structured multisignature scheme from bilinear pairing. In *Proc. of the 2nd International Symposium on Parallel and Distributed Processing and Applications (ISPA 2004), Hong Kong, China*, volume 3358 of *Lecture Notes in Computer Science*, pages 705–714. Springer-Verlag, December 2004.

[62] Chih-Yin Lin, Tzong-Chen Wu, and Fangguo Zhang. A structured multisignature scheme from the gap diffie-hellman group, May 2003. Cryptology ePrint Archive: Listing for 2003.

[63] Joseph K. Liu, Man Ho Au, and Willy Susilo. Self-generated certificate public key cryptography and certificateless signature / encryption scheme. In *Proc. of the 2nd ACM symposium on Informatoin, Computer and Communications Security (ASIACCS 2007), Singapole*, pages 273–283. ACM, March 2007.

[64] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracle. In *Proc. of the 25th Theory and Applications of Cryptographic Techniques (EUROCRYPT 2006), Petersburg, Russia*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer-Verlag, May 2006.

[65] Ben Lynn. Pbc library, 2013. `http://crypto.stanford.edu/pbc/`.

[66] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In *Proc. of the 23th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004), Interlaken, Switzerland*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90. Springer-Verlag, May 2004.

[67] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *Proc. of the 8th ACM Conference on Computer and Communications Security (CCS 2001), Philadelphia, USA*, pages 245–254. ACM, November 2001.

[68] Shirow Mitomi and Atsuko Miyaji. A general model of multisignature schemes with message flexibility,. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(10):2488–2499, October 2001.

[69] Murtaza Motiwala, Robert Lychev, Adam O'Neill, Nick Feamster, and Andy Bavier. In-band network fault localization, 2010. `http://gtnoise.net/papers/submission/orchid2010.pdf`.

[70] David Naccache. Secure and practical identity-based encryption, October 2005. Cryptology ePrint Archive: Listing for 2005, `http://eprint.iacr.org/2005/369`.

[71] Gregory Neven. Efficient sequential aggregate signed data. *IEEE Transactions on Information Theory*, 57(3):1803–1815, March 2011.

[72] Motoji Ohmori, Eikoh Chida, Hiroki Shizuya, and Takao Nishizeki. A note on the multisignature over a non-commutative ring. Technical report, IEICE, March 1996. Only in Japanese.

[73] Kazuo Ohta and Tatsuaki Okamoto. Multi-signature schemes secure against active insider attacks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1):21–31, January 1999.

[74] Tatsuaki Okamoto. A digital multisignature scheme using bijective public-key cryptosystems. *ACM Transaction on Computer Systems*, 6(4):432–441, 1988.

[75] OpenID. Connect — openid. `http://openid.net/connect/`.

[76] Jaroslaw Pastuszak, Dariusz Michalek, Josef Pieprzyk, and Jennifer Seberry. Identification of bad signatures in batches. In *Proc. of the 3rd International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2000), Melbourne, Australia*, volume 1751 of *Lecture Notes in Computer Science*, pages 28–45. Springer-Verlag, January 2000.

[77] Kenneth G. Paterson and Geraint Price. A comparison between traditional public key infrastructures and identity-based cryptography. *Information Security Technical Report*, 8(3):57–72, 2003.

[78] Geraint Price and Chris J. Mitchell. Interoperation between a conventional pki and an id-based infrastructure. In *Proc. of the Second European PKI Workshop: Research and Applications (EuroPKI 2005), Canterbury, UK*, volume 3545 of *Lecture Notes in Computer Science*, pages 73–85. Springer-Verlag, July 2005.

[79] Yakov Rekhter and Tony Li. A border gateway protocol 4 (bgp-4). RFC 1771, March 1995. http://www.ietf.org/rfc/rfc1771.txt.

[80] RIPE. Youtube hijacking: A ripe ncc ris case study, 2008. `http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study`.

[81] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attack. In *Proc. of the 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2007), Barcelona, Spain*, volume 4515 of *Lecture Notes in Computer Science*, pages 228–245. Springer-Verlag, May 2007.

[82] Markus Rückert and Dominique Schröder. Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In *Proc. of the 3rd International Conference on Information Security and Assurance (ISA 2009), Seoul, Korea*, volume 5576 of *Lecture Notes in Computer Science*, pages 750–759. Springer-Verlag, June 2009.

[83] Tatsuhiko Sano, Yoshio Kakizaki, Masaki Inamura, and Keiichi Iwamura. Implementation and evaluation of a content editing system enabling pre-control for content circulation. In *Proc. of the 30th Symposium on Cryptography and Information Security (SCIS 2013), Kyoto, Japan*. IEICE, January 2013. Japanese.

[84] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[85] Dominique Schröder. How to aggregate the cl signature scheme. In *Proc. of the 16th European Symposium on Research in Computer Security (ESORICS 2011) , Leuven, Belgium*, volume 6879 of *Lecture Notes in Computer Science*, pages 298–314. Springer-Verlag, September 2011.

[86] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of International Cryptograpy Conference (CRYPTO 1984), Santa Barbara, USA*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, August 1987.

[87] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proc. of the 13th International Conference on Practice and Theory in Public Key Cryptography (PKC 2010), Paris, France*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer-Verlag, May 2010.

[88] Philip Smith. Bgp multihoming techniques, October 2007. NANOG 41.

[89] Kotikalapudi Sriram, Oliver Borchert, Okhee Kim, David Cooper, and Doug Montgomery. Rib size estimation for bgpsec, 2011. `http://www.antd.nist.gov/~ksriram/BGPSEC_RIB_Estimation.pdf`.

[90] Mitsuru Tada. A secure multisignature scheme with signing order verifiability. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(1):73–88, January 2003.

[91] Francisco Valera, Iljitsch Van Beijnum, Alberto Garcia-Martinez, and Marcelo Bagnulo. *Multi-Path BGP: Motivations and Solutions*, chapter 1, pages 238–256. Cambridge University Press, 2011.

[92] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proc. of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2010), French Riviera*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer-Verlag, May 2010. Full paper is available in `http://eprint.iacr.org/2009/616`.

[93] Scott Vanstone. Responses to nist's proposal. *Communications of the ACM*, 35:50–52, July 1992.

[94] Lihua Wang, Eiji Okamoto, Ying Miao, Takeshi Okamoto, and Hiroshi Doi. An id-sp-m4m scheme and its security analysis. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E90-A(1):91–100, January 2007.

[95] Brent Waters. Efficient identity-based encryption without random oracles. In *Proc. of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2005), Aarhus, Denmark*, volume 3494 of *Lecture Notes oi Computer Science*, pages 114–127. Springer-Verlag, May 2005.

[96] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Proc. of the 29th Annual International Cryptology Conference (CRYPTO 2009), Santa Barbara, USA*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer-Verlag, August 2009.

[97] Chenhuang Wu. Self-generated-certificate digital signature. In *Proc. of the 4th International Conference on Generic and Evolutionary Computing (ICGEC 2010), Shenzhen, China*, pages 379–382. IEEE, December 2010.

[98] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. Certificate-based signatures revisited. *Journal of Universal Computer Science*, 15(8):1659–1684, August 2009.

[99] Hu Xiong, Qianhong Wu, and Zhong Chen. Strong security enabled certificateless aggregate signatures applicable to mobile computation. In *Proc. of the 3rd International Conference on Intelligent Networking and Collaborative Systems (INCoS 2011), Fukuoka, Japan*, pages 92–99. IEEE, November-December 2011.

[100] Dan Yamamoto and Wakaha Ogata. A general model of structured multisignatures with message flexibility. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E90-A(1):83–90, January 2007.

[101] Liang Yan, Chunming Rong, and Gansen Zhao. Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography. In *Proc. of the First International Conference (CloudCom 2009), Beijing, China*, volume 5931 of *Lecture Notes in Computer Science*, pages 167–177. Springer-Verlag, December 2009.

[102] Naoto Yanai, Eiji Chida, and Mambo Mambo. A secure structured multisignature scheme based on a non-commutative ring homomorphism. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E94-A(6):1346–1355, June 2011.

[103] Naoto Yanai, Raylin Tso, Masahiro Mambo, and Eiji Okamoto. Certificateless ordered sequential aggregate signature scheme. In *Proc. of the 3rd International Conference on Intelligent Networking and Collaborative Systems (INCoS 2011), Fukuoka, Japan*, pages 662–667. IEEE, November-December 2011.

[104] Youtube. `http://www.youtube.com/`.

[105] Lei Zhang, Bo Qin, Qianhong Wu, and Futai Zhang. Efficient many-to-one authentication with certificateless aggregate signatures. *Computer Networks*, 54(14):2482–2491, October 2010.

[106] Lei Zhang and Futai Zhang. Security model for certificateless aggregate signature schemes. In *Proc. of the International Conference on Computational Intelligence (ICCIS 2008), Zuzhou, China*, pages 364–368. IEEE, December 2008.

[107] Lei Zhang and Futai Zhang. A new certificateless aggregate signature scheme. *Computer Communications*, 32(6):1079–1085, April 2009.

[108] Meiyuan Zhao, Sean Smith, and David Nicol. Aggregated path authentication for efficient bgp security. In *Proc. of the 12th ACM Conference on Computer and Communications Security (CCS 2005), Alexandria, USA*, pages 128–138. ACM, November 2005.

Publication List

Journals

- Naoto Yanai, Eikoh CHIDA, and Masahiro MAMBO, "A Secure Structured Multisignature Scheme Based on a Non-Commutative Ring Homomorphism", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E94-A, No.6, pp.1346-1355, June 2011.

- Naoto Yanai, Raylin Tso, Masahiro MAMBO, and Eiji Okamoto, "A Certificateless Ordered Sequential Aggregate Signature Scheme Secure against Super Adversaries", Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications, Vol.3, No. 1/2, pp.30-54, March 2012.

- Naoto Yanai, Eikoh CHIDA, Masahiro MAMBO, "A CDH-based Ordered Multisignature Scheme Provably Secure without Random Oracles", Journal of Information Processing, Recommended Paper, Vol.22, No. 2, to appear.

International Conference

- Naoto Yanai, Eikoh CHIDA, and Masahiro MAMBO, "A Structured Multisignature Based on a Non-Commutative Ring Homomorphism", Proc. of the firth Joint Workshop on Information Security (JWIS) 2009, 1B-3, pp.1-15, March 2009.

- Naoto Yanai, Eikoh CHIDA, and Masahiro MAMBO, "A Structured Aggregate Signature Scheme", Proc. of the 2010 International Symposium on Information Theory and its Applications (ISITA 2010), pp.795-800, October 2010.

- Naoto Yanai, Raylin Tso, Masahiro MAMBO, and Eiji Okamoto, "Certificateless Ordered Sequential Aggregate Signature Scheme", Proc. of the third International Conference on Intelligent Networking and Collaborative Systems (INCoS) 2011, (Workshop: Third International Workshop on Managing Insider Security Threats (MIST) 2011), pp.662-667, November - December 2011.

- Naoto Yanai, Masahiro MAMBO, and Eiji Okamoto, "Ordered Multisignature Schemes under the CDH Assumption without Random Oracles", Pre-proc. of the 16th Information Security Conference (ISC) 2013, November 2013. To appear in the final proc. as a volume of Lecture Notes in Computer Science in 2014.

Invited Talk

- Naoto Yanai, "Multiple-Signability: A Study on Multisignatures and Aggregate Signatures", the 8th International Workshop on Security (IWSEC) 2013, SCIS/CSS Invited Sessions, November 2013.

Domestic Symposium

- 矢内直人, 千田 栄幸, "権限の違いを考慮した多重署名方式", 情報処理学会東北支部研究会, 平成 20 年度 第 6 回, A-3-5, pp.1-8, 2009 年 3 月.

- 矢内 直人, 千田 栄幸, 満保 雅浩, "署名順序を検証可能な確率的 ID ベースアグリゲート署名方式", 2011 年 暗号と情報セキュリティシンポジウム (SCIS), 3A1-1, pp.1-6, 2011 年 1 月.

- 矢内 直人, 千田 栄幸, 満保 雅浩, "順序検証可能な多重署名方式の安全性証明に関する一考察", 2011 年 暗号と情報セキュリティシンポジウム (SCIS), 3A1-2, pp.1-6, 2011 年 1 月.

- 矢内 直人, 秋山 浩岐, 小久保 博崇, 窪田 峻, 太田 佳樹, "構造型集約多重署名方式の開発", コンピュータサイエンス専攻テクニカルレポート, ソリューション型研究開発プロジェクト 2010 年度研究成果報告, pp.113-118, 2011 年 2 月.

- Naoto YANAI, Raylin TSO, Masahiro MAMBO, and Eiji OKAMOTO, "On the Security of Certificateless Ordered Sequential Aggregate Signature Scheme against Super Adversaries", 2012 年暗号と情報セキュリティシンポジウム (SCIS), 4A2-2, pp.1-8, 2012 年 1 月.

- 矢内 直人, 千田 栄幸, 満保 雅浩, 岡本 栄司, “スタンダードモデルにおける順序検証型多重署名方式”, 2012 年 コンピュータセキュリティシンポジウム (CSS), 2C1-3, pp.1-8, 2012 年 10 月. CSS 2012 学生論文賞受賞.
- 矢内 直人, 千田 栄幸, 満保 雅浩, 岡本 栄司, “多者多重署名の構成に関する一考察”, 2013 年 暗号と情報セキュリティシンポジウム (SCIS), 3A4-3, pp.1-8, 2013 年 1 月. SCIS 論文賞受賞.
- 矢内 直人, 千田 栄幸, 満保 雅浩, 花岡 悟一郎, 岡本栄司, “多者多重署名とその応用”, 第 12 回情報科学技術フォーラム (FIT 2013), L-027, 第 4 分冊, pp.281-282, 2013 年 9 月.
- 矢内 直人, 千田 栄幸, 満保 雅浩, 岡本 栄司, “BGP 指向アグリゲート署名の構成”, 2013 年 コンピュータセキュリティシンポジウム (CSS), 2C3-1, pp.510-517, 2013 年 10 月.
- Naoto Yanai, Eikoh Chida, Masahiro Mambo, and Eiji Okamoto, “Efficient CD-based Ordered Multisignature Schemes without Random Oracles”, 第 36 回情報理論とその応用シンポジウム (SITA 2013), pp.134-139, 2013 年 11 月.