A STRUCTURED AGGREGATE SIGNATURE SCHEME WITH PAIRING-BASED CRYPTOGRAPHY

Masaki Inamura and Keiichi Iwamura
Department of Electrical Engineering, Graduate School of Engineering, Tokyo University of Science.
1-14-6 Kudankita, Chiyoda-ku, Tokyo, 102-0073 Japan.
minamura@sec.ee.kagu.tus.ac.jp, iwamura@ee.kagu.tus.ac.jp

ABSTRACT
In this study, we propose a new aggregate signature scheme with pairing-based cryptography that can describe the mixed/parallel structure of signers. Existing structured signature schemes are based on multisignature schemes, which are adapted for verification of a mixed/parallel structure if all signers sign the same document. However, if each signer wants to sign a document different from those of other signers, another scheme that is based on an aggregate signature scheme is required. To resolve the above problem, a denoted connective signature is generated in our scheme. In addition, our scheme is shown to be secure under the Gap-Diffie-Hellman assumption. Furthermore, we explain that our scheme is simple to construct, and its performance is efficient.

KEYWORDS
Signature, Aggregate signature, Pairing, Gap Diffie-Hellman, Elliptic curve

1 INTRODUCTION

Recently, Internet has become more commonly used, and it has become possible for users to do their work more efficiently. Document distribution or grouping systems for document sharing is one example of such improved efficiency that can be realized. For systems that handle work flow, it is important for managers to confirm the receiver/sender of documents. Therefore, an effective method is required to make those checks.

For this purpose, the multisignature scheme [1] has been identified as one of the promising mechanisms. Under this scheme, each member sequentially signs an objective document, and all members' signatures are aggregated into one signature called a multisignature. A verifier can then verify the validity of all members' signatures by verifying only the multisignature. Therefore, the signing cost is effectively minimized. Furthermore, a structured multisignature scheme [2, 3] is one of the best mechanisms and describes not only the member who signs the document but also the route that can be taken by the document using only multisignature verification. These multisignature schemes are used when all signers are signing the same document. On the other hand, if every signer signs a document different from those of other signers, e.g., the addition of information data to the distributed document because of alterations made to the document by each relay signer, an aggregate signature scheme is required. However, existing aggregate signature schemes do not extend to structured schemes.

In this study, authors propose a structured aggregate signature scheme with a pairing-based cryptography called BLS signature [4]. Our contributions are described as follows:

- We propose an aggregate signature scheme that can express the order-specified structure of signers.
- We verify that our proposed aggregate signature scheme is provably secure under the Gap-Diffie-Hellman (GDH) assumption.
- We expand our proposed scheme as an aggregate signature scheme to describe parallel/mixed route defined by Lin et al [5].

We also implement a prototype and evaluate its computational performance.

2 RELATED WORKS

2.1 BLS Signature Scheme

Okamoto and Pointcheval defined a GDH class [6]. Consider a multiplicative cyclic group $\mathbb{G}'$ with a prime order p. They defined two problems for Diffie-Hellman (DH) as follows:

The computational DH (CDH) problem:
For $a, b \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}'$, given (g, g^a, g^b), compute g^{ab}.

The decisional DH (DDH) problem:
For $a, b, c \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}'$, given (g, g^a, g^b, g^c), decide whether $c \stackrel{?}{=} ab$.

Regarding the GDH class, it is defined such that the DDH problem is easy, whereas the computational DH (CDH) problem is hard.

The first example involving the use of such groups was given by Joux and Nguyen [7, 8], whereas Boneh, Lynn and Shacham showed that a new signature scheme can be structured on the basis of the GDH class, by using pairing over elliptic curves [4]. Pairing is one of the functions used on specified elliptic curves. Consider a group $\mathbb{G}$ on elliptic curves that enables pairing. Let e denote the symbol for the pairing function. Pairing has the following characteristics:

- For $P_1, P_2, Q \in \mathbb{G}$,
  then $e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)$.
- For $P, Q_1, Q_2 \in \mathbb{G}$,
  then $e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2)$.
- For $a, b \in \mathbb{Z}_p^*$ and $P, Q \in \mathbb{G}$,
  then $e(aP, bQ) = e(bP, aQ) = e(abP, Q)$
  $= e(P, abQ) = e(P, Q)^{ab}$.

Therefore, a signature scheme based on a group $\mathbb{G}$ can be structured as follows:

Key Generation: The term g is a generator of the group $\mathbb{G}$. The secret key of the signer is a random element $x \in \mathbb{Z}_p^*$, whereas his public key is $v = xg$.

Signing: H is a one-way hash function, which outputs a random element in the whole group $\mathbb{G}$, e.g., *MapToGroup* hashing onto $\mathbb{G}^*$ [4]. The term m is both a plain message and signing target. The signer computes $h = H(m)$ and returns $\sigma = xh$.

Verification: When the verifier is provided values of g, v, m and σ, he/she computes $h = H(m)$ and verifies $e(g, \sigma) \stackrel{?}{=} e(v, h)$.

While verifying, if a signer generates a signature accurately, a verifier can verify this signature using the pairing result as follows:

$$e(g, \sigma) = e(g, xh) = e(g, h)^x.$$
$$e(v, h) = e(xg, h) = e(g, h)^x.$$

Therefore, if $e(g, \sigma) = e(v, h)$, then σ has been generated accurately. Furthermore,it has been shown that the above scheme is secure against chosen-message attacks [4]. In this study, we call this signature scheme that is structured on the group $\mathbb{G}$ with pairing *the BLS signature scheme*.

After proposing the concrete GDH signature scheme, Boneh et al. also proposed also a prototype of the aggregate signature scheme on the basis of *the BLS signature scheme* [9].Consider a group $\mathbb{U} = \{u_1, \ldots, u_n\}$, which is a group of n signers, and a subgroup $\mathbb{L} = \{u_{i_1}, \ldots, u_{i_l}\} \subseteq \mathbb{U}$, which is a group of l members who actually participate in an aggregate signature. Let $\mathbb{J} = \{i_1, \ldots, i_l\}$ denote the set of indices of such members. Then, proposers showed that we could structure the aggregate signature scheme on the basis of *the BLS signature scheme* as follows:

Key Generation: The term g is a generator of the group $\mathbb{G}$. The secret key of the signer $u_i \in \mathbb{U}$ is a random element $x_i \in \mathbb{Z}_p^*$, whereas his public key is $v_i = x_i g$.

Signing: H is a one-way hash function that outputs a random element in the whole group $\mathbb{G}$. The term m_{i_α} is both a plain message and signing target of the signer $u_{i_\alpha} \in \mathbb{L}$. The signer u_{i_α} computes $h_{i_\alpha} = H(m_{i_\alpha})$ and returns $\sigma_{i_\alpha} = x_{i_\alpha} h_{i_\alpha}$.

Aggregation: The issuer of an aggregate signature finally collects all σ_{i_α} generated by u_{i_α}, computes $\sigma = \sum_{j \in \mathbb{J}} \sigma_j$ and returns $(m_{i_1}, \ldots, m_{i_l}, \mathbb{L}, \sigma)$.

Verification: When the verifier is provided values of $g, m_{i_1}, \ldots, m_{i_l}, \mathbb{L}$ and σ, he/she collects all v_{i_α} using $\mathbb{L}$, computes all $h_{i_\alpha} = H(m_{i_\alpha})$, and verifies $e(g, \sigma) \stackrel{?}{=} \prod_{j \in \mathbb{J}} e(v_j, h_j)$.

While verifying, if all signers accurately generate an aggregate signature, a verifier can verify this signature using the pairing result as follows:

$$e(g,\sigma) = e(g, \sum_{j\in\mathbb{J}} x_j h_j) = \prod_{j\in\mathbb{J}} e(g, x_j h_j) = \prod_{j\in\mathbb{J}} e(g,h_j)^{x_j}.$$

$$\prod_{j\in\mathbb{J}} e(v_j,h_j) = \prod_{j\in\mathbb{J}} e(x_j g,h_j) = \prod_{j\in\mathbb{J}} e(g,h_j)^{x_j}.$$

Therefore, if $e(g,\sigma) = \prod_{j\in\mathbb{J}} e(v_j,h_j)$, then all σ_{i_α} have been generated accurately. Furthermore, it has been proven that the above scheme is secure even with the reduced security of *the BLS signature scheme* [9].

2.2 Multisignature Protocol Considering Hierarchical Relation

Several multisignature schemes considering hierarchical relations have been discussed.

First, an order-specified multisignature scheme, which can verify the order in which signers have signed a document, is proposed [2, 3]. However, this scheme cannot distinguish between divisions to which several signers belong.

Lin et al. proposed the concept of a hierarchical multisignature scheme, and they discussed requirements for distinguishing between each hierarchy [5]. Furthermore, authors' group proposed a realistic multisignature scheme, which comprises a hierarchical structure of signers [10].In this scheme, authors' middle keys are deliberately introduced to express the hierarchical relation among divisions to which each signer belonged as well as a verification protocol using these middle keys. Therefore, they successfully verified the structure of the hierarchy.

3 ORDER-SPECIFIED AGGREGATE SIGNATURE (PROPOSITION 1)

In Section 1, we mentioned that we propose the structured aggregate signature to enable the addition of information to distributed documents.

In this section, we propose a new order-specified aggregate signature scheme based on *the BLS signature scheme* (Proposition 1); then, we propose the structured aggregate signature (Proposition 2).

3.1 Symbols, Preconditions and Requirements

3.1.1 Symbols

We define symbols used in our protocol as follows:

$\mathbb{G}_1, \mathbb{G}_2$**:** A group of elliptic curves enabling pairing.

g**:** A generator in the group $\mathbb{G}_1$.

e**:** A pairing function.

u_o**:** A signer in our order-specified aggregate signature (o is the number of order, and $u_o \neq \mathrm{null}$) .

x_o, v_o**:** A private key ($x_o \in \mathbb{Z}_p^*$) and a public key ($v_o = x_o g$), respectively, which the signer u_o holds.

$\mathbb{L}_o$**:** Order information from u_1 to u_o.

m_o**:** A plain message for u_o.

H**:** A one-way hash function onto $\mathbb{G}_2$ (e.g. *MapToGroup*).

σ_o**:** An order-specified aggregate signature from u_1 to u_o.

3.1.2 Preconditions

In our proposal, we assume the following preconditions:

- Our order-specified aggregate signature scheme is operated with a legal certification authority (CA).
- While signing, all signers accurately generate $\mathbb{L}_o$ and σ_o.
- No other key-pairs without them that are legally generated for signers are produced.
- While signing, because of the security on networks, third parties cannot wire-tap.

3.1.3 Requirements

Komano et al. showed the need to define security requirements for an aggregate signature scheme that is different from those for a standard digital signature scheme [11,12]. Therefore, we define security requirements for our scheme as follows:

(1) Unforgeability

Although a malicious third party can obtain all public information, it cannot generate σ_n. This requirement also fulfills order-specified legitimacy.

(2) Collusion-secure

A signer cannot collude with another signer for denial of aggregate signature participation.

(3) Impossibility of Deceptive Participation

A signer cannot allow a third party to participate in an aggregate signature without permission.

3.2 Protocol

3.2.1 Key Pair Generation

The term g is a generator of the group $\mathbb{G}_1$. The secret key of the signer u_i is a random element, x_i (however, if $i \neq j$ then $x_i \neq x_j$), whereas his public key is $v_i = x_i g$.

When the signer u_i generates the key pair, u_i sends the public key to CA, who generates a random value $r \in \mathbb{Z}_p^*$ and sends it to u_i. The signer u_i signs r using the secret key with *the BLS signature scheme* and sends the signature to CA, who verifies the signature using the public key. If verification is successful, CA registers the public key.

3.2.2 Signing and Aggregation

Here, we show the procedure for signing and aggregation as follows:

S1. The signer u_1, who is first in the order, computes the following:

$$\begin{aligned} h_1 &= H(m_1). \\ \sigma_1 &= x_1 h_1. \\ \mathbb{L}_1 &= \{(\text{null}, u_1)\}. \end{aligned}$$

This signer sends the signature set $(m_1, \sigma_1, \mathbb{L}_1)$ to a directly-connected signer in the second order.

S2. The signer u_i, who is i in the order, computes the following:

$$\begin{aligned} h_i &= H(m_i). \\ \sigma_i &= \sigma_{i-1} + x_i h_{i-1} + x_i h_i \\ &= x_1 h_1 + \sum_{j=2}^{i} (x_j h_{j-1} + x_j h_j). \\ \mathbb{L}_i &= \mathbb{L}_{i-1} + \{(u_{i-1}, u_i)\} \\ &= \{(\text{null}, u_1), (u_1, u_2), \ldots, (u_{i-1}, u_i)\}. \end{aligned}$$

This signer sends the signature set $(m_1, \ldots, m_i, \sigma_i, \mathbb{L}_i)$ to a directly-connected signer in the order $i+1$. This procedure is reflexively executed until the order $n-1$.

S3. The signer u_n, who is last in the order, computes the following:

$$\begin{aligned} h_n &= H(m_n). \\ \sigma_n &= \sigma_{n-1} + x_n h_{n-1} + x_n h_n \\ &= x_1 h_1 + \sum_{j=2}^{n} (x_j h_{j-1} + x_j h_j). \\ \mathbb{L}_n &= \mathbb{L}_{n-1} + \{(u_{n-1}, u_n)\} \\ &= \{(\text{null}, u_1), (u_1, u_2), \ldots, (u_{n-1}, u_n)\}. \end{aligned}$$

This signer returns the signature set $(m_1, \ldots, m_n, \sigma_n, \mathbb{L}_n)$ as the order-specified aggregate signature set.

3.2.3 Verification

When the verifier is provided the order-specified aggregate signature set and has a mean to obtain all signers' public keys, he or she verifies the aggregate signature as follows:

V1. The verifier collects all v_i using $\mathbb{L}_n$.

V2. The verifier computes $h_i = H(m_i)$.

V3. The verifier verifies the following:

$$e(v_1, h_1)(\prod_{j=2}^{n} e(v_j, h_{j-1} + h_j)) \stackrel{?}{=} e(g, \sigma_n).$$

While verifying, if all signers accurately generate an aggregate signature, a verifier can verify this signature

with the pairing result as follows:

$$
\begin{aligned}
& e(v_1, h_1) \cdot (\prod_{j=2}^{n} e(v_j, h_{j-1} + h_j)) \\
=& e(v_1, h_1) \cdot e(v_2, h_1 + h_2) \ldots e(v_n, h_{n-1} + h_n) \\
=& e(x_1 g, h_1) \cdot e(x_2 g, h_1 + h_2) \ldots e(x_n g, h_{n-1} + h_n) \\
=& e(g, x_1 h_1) \cdot e(g, x_2(h_1 + h_2)) \\
& \qquad \ldots e(g, x_n(h_{n-1} + h_n)) \\
=& e(g, x_1 h_1 + \sum_{j=2}^{n} (x_j h_{j-1} + x_j h_j)) \\
=& e(g, \sigma_n).
\end{aligned}
$$

4 NEW STRUCTURED AGGREGATE SIGNATURE (PROPOSITION 2)

Our scheme in Section 3 expresses order connection by repeating procedure S1, in which two directly-connected signers' documents are signed by the latter signer. This expression is not limited to one-to-one connections but enables the expansion into one-to-many connections. Therefore, we propose a structured aggregate signature by repeating the connection between two directly-connected signers.

In this section, we explain our new structured aggregate signature scheme. As a sample case, we consider our system on signing structures, which are the same as those in the study of Lin et al [5]. These structures are shown in Figure 1.

4.1 Protocol for the Parallel Structure

4.1.1 Key Pair Generation

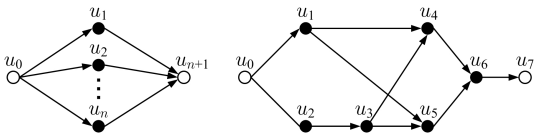
The procedure is same as that mentioned in Section 3.2.1.

4.1.2 Signing and Aggregation

Here, we show the procedure for signing and aggregation as follows:

PS1. The signer u_0, who is first in the order, computes the following:

$$
\begin{aligned}
h_0 =& H(m_0). \\
\sigma_0 =& x_0 h_0. \\
\mathbb{L}_0 =& \{(\text{null}, u_0)\}.
\end{aligned}
$$



4.1 Parallel Structure 4.2 Mixed Structure (example)

Figure 1: Signing Structure.

This signer sends the signature set $(m_0, \sigma_0, \mathbb{L}_0)$ to next signers $u_1, \ldots, u_n$.

PS2. The signer u_i $(1 \leq i \leq n)$, who is second in the order, computes the following:

$$
\begin{aligned}
h_i =& H(m_i). \\
\sigma_i =& \sigma_0 + x_i h_0 + x_i h_i \\
=& x_0 h_0 + x_i h_0 + x_i h_i. \\
\mathbb{L}_i =& \mathbb{L}_0 + \{(u_0, u_i)\} \\
=& \{(\text{null}, u_0), (u_0, u_i), \}.
\end{aligned}
$$

This signer u_i sends the signature set $(m_0, m_i, \sigma_0, \sigma_i, \mathbb{L}_i)$ to the last signer u_{n+1}. This procedure is executed by all signers u_i.

PS3. The signer u_{n+1}, who is last in the order, computes the following:

$$
\begin{aligned}
h_{n+1} =& H(m_{n+1}). \\
\sigma_{n+1} =& \sum_{j=1}^{n} (\sigma_j + x_{n+1} h_j) \\
& - (n-1)\sigma_1 + x_{n+1} h_{n+1} \\
=& x_0 h_0 \\
& + \sum_{j=1}^{n} ((x_j h_0 + x_j h_j + x_{n+1} h_j)) \\
& + x_{n+1} h_{n+1}. \\
\mathbb{L}_n =& \sum_{j=1}^{n} \mathbb{L}_j + \{(u_j, u_{n+1})\} \\
=& \{(\text{null}, u_0), (u_0, u_1), \ldots, (u_0, u_n), \\
& \qquad (u_1, u_{n+1}), \ldots, (u_1, u_{n+1})\}.
\end{aligned}
$$

This signer returns the signature set $(m_0, \ldots, m_{n+1}, \sigma_{n+1}, \mathbb{L}_{n+1})$ as the parallel-structure-specified aggregate signature set.

4.1.3 Verification

When the verifier is given the parallel-structure-specified aggregate signature set and has a mean to obtain all signers' public keys, he or she verifies the aggregate signature as follows:

PV1. The verifier collects all $v_0, \ldots, v_{n+1}$ using $\mathbb{L}_{n+1}$.

PV2. The verifier computes
$h_0 = H(m_0), \ldots, h_{n+1} = H(m_{n+1})$.

PV3. The verifier defines the parallel structure using $\mathbb{L}_{n+1}$ and verifies the following:

$$\begin{aligned} & e(v_0, h_0) \cdot (\prod_{j=1}^{n} e(v_j, h_0 + h_j)) \\ & \cdot (\prod_{j=1}^{n} e(v_{n+1}, h_j)) \cdot e(v_{n+1}, h_{n+1}) \\ \stackrel{?}{=} & e(g, \sigma_{n+1}). \end{aligned}$$

While verifying, if all signers accurately generate an aggregate signature, a verifier can verify this signature using the pairing result as follows:

$$\begin{aligned} & e(v_0, h_0) \cdot (\prod_{j=1}^{n} e(v_j, h_0 + h_j)) \\ & \cdot (\prod_{j=1}^{n} e(v_{n+1}, h_j)) \cdot e(v_{n+1}, h_{n+1}) \\ = & e(x_0 g, h_0) \cdot (\prod_{j=1}^{n} e(x_j g, h_0 + h_j)) \\ & \cdot (\prod_{j=1}^{n} e(x_{n+1} g, h_j)) \cdot e(x_{n+1} g, h_{n+1}) \\ = & e(g, x_0 h_0) \cdot (\prod_{j=1}^{n} e(g, x_j (h_0 + h_j))) \\ & \cdot (\prod_{j=1}^{n} e(g, x_{n+1} h_j)) \cdot e(g, x_{n+1} h_{n+1}) \\ = & e(g, x_0 h_0 + \left(\sum_{j=1}^{n} (x_j h_0 + x_j h_j + x_{n+1} h_j) \right) \\ & \qquad + x_{n+1} h_{n+1}) \\ = & e(g, \sigma_n). \end{aligned}$$

4.2 Protocol for the Mixed Structure

In this section, we explain the protocol using the example mentioned in Figure 1.

4.2.1 Key Pair Generation

The procedure is same as that mentioned in Section 3.2.1.

4.2.2 Signing and Aggregation

Here, we show the procedure for signing and aggregation as follows:

MS1. The signer u_0, who is first in the order, computes the following:

$$\begin{aligned} h_0 =& H(m_0). \\ \sigma_0 =& x_0 h_0. \\ \mathbb{L}_0 =& \{(\text{null}, u_0)\}. \end{aligned}$$

This signer sends the signature set $(m_0, \sigma_0, \mathbb{L}_0)$ to next signers u_1 and u_2.

MS2. The signer u_1 computes the following:

$$\begin{aligned} h_1 =& H(m_1). \\ \sigma_1' =& x_1 h_1. \\ \sigma_1 =& \sigma_0 + x_1 h_0 + x_1 h_1. \\ \mathbb{L}_0 =& \{(\text{null}, u_0), (u_0, u_1)\}. \end{aligned}$$

This signer sends the signature set $(m_0, m_1, \sigma_0, \sigma_1', \sigma_1, \mathbb{L}_1)$ to next signers u_4 and u_5.

MS3. The signer u_2 computes the following:

$$\begin{aligned} h_2 =& H(m_2). \\ \sigma_2' =& x_2 h_2. \\ \sigma_2 =& \sigma_0 + x_2 h_0 + x_2 h_2. \\ \mathbb{L}_2 =& \{(\text{null}, u_0), (u_0, u_2)\}. \end{aligned}$$

This signer sends the signature set $(m_0, m_2, \sigma_0, \sigma_2', \sigma_2, \mathbb{L}_2)$ to the next signer u_3.

MS4. The signer u_3 computes the following:

$$\begin{aligned} h_3 =& H(m_3). \\ \sigma_3' =& x_3 h_3. \\ \sigma_3 =& \sigma_2 + x_3 h_2 + x_3 h_3. \\ \mathbb{L}_2 =& \mathbb{L}_1 + \{(u_2, u_3)\}. \end{aligned}$$

This signer sends the signature set $(m_0, m_2, m_3, \sigma_0, \sigma'_2, \sigma'_3, \sigma_3, \mathbb{L}_3)$ to next signers u_4 and u_5.

MS5. The signer u_4 computes the following:

$$\begin{aligned} h_4 =& H(m_4). \\ \sigma'_4 =& x_4 h_4. \\ \sigma_4 =& \sigma_1 + \sigma_3 - \sigma_0 + x_4 h_1 + x_4 h_3 + x_4 h_4. \\ \mathbb{L}_4 =& \mathbb{L}_1 + \mathbb{L}_3 + \{(u_1, u_4), (u_3, u_4)\}. \end{aligned}$$

This signer sends the signature set $(m_0, m_1, m_2, m_3, m_4, \sigma_0, \sigma'_1, \sigma'_2, \sigma'_3, \sigma'_4, \sigma_4, \mathbb{L}_4)$ to the next signer u_6.

MS6. The signer u_5 computes the following:

$$\begin{aligned} h_5 =& H(m_5). \\ \sigma'_5 =& x_5 h_5. \\ \sigma_5 =& \sigma_1 + \sigma_3 - \sigma_0 + x_5 h_1 + x_5 h_3 + x_5 h_5. \\ \mathbb{L}_5 =& \mathbb{L}_1 + \mathbb{L}_3 + \{(u_1, u_5), (u_3, u_5)\}. \end{aligned}$$

This signer sends the signature set $(m_0, m_1, m_2, m_3, m_5, \sigma_0, \sigma'_1, \sigma'_2, \sigma'_3, \sigma'_5, \sigma_5, \mathbb{L}_5)$ to the next signer u_6.

MS7. The signer u_6 computes the following:

$$\begin{aligned} h_6 =& H(m_6). \\ \sigma'_6 =& x_6 h_6. \\ \sigma_6 =& \sigma_4 + \sigma_5 - \sigma_0 - \sigma' 1 - \sigma' 3 \\ & + x_6 h_4 + x_6 h_5 + x_6 h_6. \\ \mathbb{L}_6 =& \mathbb{L}_4 + \mathbb{L}_5 + \{(u_4, u_6), (u_5, u_6)\}. \end{aligned}$$

This signer sends the signature set $(m_0, \ldots, m_6, \sigma_0, \sigma'_1, \ldots, \sigma'_6, \sigma_6, \mathbb{L}_6)$ to the next signer u_6.

MS8. The signer u_7, who is last in the order, computes the following:

$$\begin{aligned} h_7 =& H(m_7). \\ \sigma_7 =& \sigma_6 + x_7 h_6 + x_7 h_7. \\ \mathbb{L}_7 =& \mathbb{L}_6 + \{(u_6, u_7)\}. \end{aligned}$$

This signer returns the signature set $(m_0, \ldots, m_7, \sigma_7, \mathbb{L}_7)$ as the mixed-structure-specified aggregate signature set.

4.2.3 Verification

When the verifier is provided the mixed-structure-specified aggregate signature set and has a mean to obtain all signers' public keys, he or she verifies the aggregate signature as follows:

MV1. The verifier collects all $v_0, \ldots, v_7$ using $\mathbb{L}_7$.

MV2. The verifier computes
$h_0 = H(m_0), \ldots, h_7 = H(m_7)$.

MV3. The verifier defines the mixed structure using $\mathbb{L}_7$ and verifies the following:

$$\begin{aligned} & e(v_0, h_0) \cdot e(v_1, h_0 + h_1) \cdot e(v_2, h_0 + h_2) \\ & \cdot e(v_3, h_2 + h_3) \cdot e(v_4, h_1 + h_3 + h_4) \\ & \cdot e(v_5, h_1 + h_3 + h_5) \\ & \cdot e(v_6, h_4 + h_5 + h_6) \cdot e(v_7, h_6 + h_7) \\ \stackrel{?}{=} & e(g, \sigma_7). \end{aligned}$$

While verifying, if all signers accurately generate an aggregate signature, a verifier can verify this signature using the pairing result as follows:

$$\begin{aligned} & e(v_0, h_0) \cdot e(v_1, h_0 + h_1) \cdot e(v_2, h_0 + h_2) \\ & \cdot e(v_3, h_2 + h_3) \cdot e(v_4, h_1 + h_3 + h_4) \\ & \cdot e(v_5, h_1 + h_3 + h_5) \\ & \cdot e(v_6, h_4 + h_5 + h_6) \cdot e(v_7, h_6 + h_7) \\ =& e(x_0 g, h_0) \cdot e(x_1 g, h_0 + h_1) \cdot e(x_2 g, h_0 + h_2) \\ & \cdot e(x_3 g, h_2 + h_3) \cdot e(x_4 g, h_1 + h_3 + h_4) \\ & \cdot e(x_5 g, h_1 + h_3 + h_5) \\ & \cdot e(x_6 g, h_4 + h_5 + h_6) \cdot e(x_7 g, h_6 + h_7) \\ =& e(g, x_0 h_0) \cdot e(g, x_1(h_0 + h_1)) \\ & \cdot e(g, x_2(h_0 + h_2)) \cdot e(g, x_3(h_2 + h_3)) \\ & \cdot e(g, x_4(h_1 + h_3 + h_4)) \\ & \cdot e(g, x_5(h_1 + h_3 + h_5)) \\ & \cdot e(g, x_6(h_4 + h_5 + h_6)) \cdot e(g, x_7(h_6 + h_7)) \\ =& e(g, \sigma_0 + \left(\sum_{j=1}^{6} sigma'_j \right) + x_7 h_7 \\ & + x_1 h_0 + x_2 h_0 + x_3 h_2 + x4(h_1 + h_3) \\ & + x_5(h_1 + h_3) + x_5(h_4 + h_5) + x_7 h_6) \\ =& e(g, \sigma_7). \end{aligned}$$

5 DISCUSSION

5.1 Security Analysis

We defined security requirements for our proposed scheme and discussed the security analysis for those requirements in Section 3.1.3.

In this study, we propose three types of schemes. However, all schemes have similar terms, which show who signs a document ($x_i h_i$) and which paired signers are directly connected ($x_i h_j$, $i \neq j$). Therefore, in Section 3, we mention the analysis of only the order-specified aggregate signature scheme.

(1) Unforgeability

An order-specified aggregate signature is given by σ_n (n is the last order number) at the end. Therefore, if a malicious third party cannot generate σ_n, the legally generated aggregate signature is accurate.

We can prove the security of unforgeability as follows:

Theorem.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be a GDH groups. Then, σ_n is secure against forgery of the aggregate signature set in the random oracle.

Proof of the Theorem.

We prove that σ_n is secure with a reduction in *the BLS signature scheme* using the proof procedure proposed by Boldyreba [13, 14]. Let A be a poly-time adversary for signing σ_n. We will structure the adversary B for *the BLS signature scheme* such that

$\mathsf{Adv}_{\sigma_n}^{\text{Aggregate-sign}}(A) = \mathsf{Adv}_{\textit{the-BLS-signature}}^{\text{sign}}(B)$.

This means that if *the BLS signature scheme* is secure, then computing σ_n is secure.

If σ_i is open to public for another aggregate signature, it is obvious that A succeeds in forgery whenever B is successful. Therefore, we prove the converse as follows:

B has a public key v_i and an access to random and signing oracles. B will run A, simulating the honest player. First, B gives public key v_1 to A as the signer's key in the last order. There should be x_1 corresponding to the secret key of v_1. However, the adversary cannot realize x_1 because of the discrete logarithm problem. Then, A outputs the set of $n-1$ key-pairs, which should be computed by signers $(x_2, v_2), \ldots, (x_n, v_n)$, where $v_i = x_i g$ $(2 \leq i \leq n)$. Let the above key pairs be signers' keys, without the first signer. B simulates A's random oracle using its own oracle. Whenever A asks the honest player to participate in the order-specified signature generation protocol for some message m_j $(1 \leq j \leq n)$, B forwards the query to its signing oracle and returns the reply to A. At some point, A outputs an attempted forgery σ_n. Then, B is computed as follows:

$$\begin{aligned}&\sigma_n - \sum_{i=2}^{n} (x_i \cdot (H(m_{i-1}) + H(m_i))) \\ =&x_1 H(m_1).\end{aligned}$$

It can be observed that B simulates A for a valid experiment, runs in time comparable to the running time of A, and succeeds in forgery whenever A is successful. (QED)

(2) Collusion-secure

The connection between two signers (u_{i-1} and u_i) is indicated by computation as follows:

$$(x_{i-1} + x_i) H(m_{i-1}).$$

If x_j and x_k $(j \neq k)$ for which
$x_j + x_k = x_{i-1} + x_i$ existed, u_{i-1} and u_i could deny participation in the aggregate signature with intrusion into other signers who may hold x_j and x_k.

As a countermeasure against the above, we consider that each secret key is generated as the sum of a secret key pair that is invariably different from the sum of the other pair. If the group of keys $\mathbb{K} \subset \mathbb{Z}_p^*$ satisfied the above condition, no signer can deny participation in the aggregate signature with a collusion attack because the sum of each key pair is equal to the individual secret key.

A group $\mathbb{W}$, for which elements become a superincreasing sequence, is an example of the group $\mathbb{K}$.

Proof that $\mathbb{W}$ is an example of the group $\mathbb{K}$.

We select any two elements w_i and w_j in $\mathbb{W}$ $(w_i < w_j)$ and compare their sum with the sum of the other pair w_k and w_l in $\mathbb{W}$ $(w_k < w_l)$. If $w_l \leq w_i$, $w_k + w_l$ is always less than w_j because of one of the characteristics of the superincreasing sequence. Furthermore, if $w_l > w_j$, $w_i + w_j$ is always less than w_l because of this characteristic. Therefore, the value of $w_i + w_j$ equal to the individual secret key, and $\mathbb{W}$ is an example of $\mathbb{K}$. (QED)

Table 1: Data size.

	Signature size	Middle key size
IIWNT11	Q_s	Q_v
Proposed scheme	Q_s	0

Furthermore, Erdös proposed an efficient construction of the group $\mathbb{K}$ [15].

(3) Impossibility of Deceptive Participation

An adversary u_z, who wants to participate in the aggregate signature deceptively, holds his/her secret key, x_o, and obtains a document, m_i, and the signature, $x_i H(m_i)$, of one legal signer u_i. If the adversary, u_z, computes $x_z H(m_i) + x_i H(m_i)$ and adds this value to the aggregate signature, deceptive participation of u_z is successful.

To protect the aggregate signature against u_z, each signer includes a part of the document of the former signer in his/her own document before signing it. Thus, we can avoid deceptive participation.

5.2 Comparison with Multisignature

The data size is calculated using the size of the signature and public key on the basis of *the BLS signature scheme*, and the performance can be calculated using the number of signers/links and frequencies of the following: the pairing calculation, scalar product, and the addition over an elliptic curve. We compare the proposed scheme with the structured multisignature scheme, IIWNT11 [10].

First, we compare the proposed scheme with IIWNT11 in the data size and show the results in Table 1. We define the following:

Q_s**:** Data size of the signature based on *the BLS signature scheme*.

Q_v**:** Data size of the public key based on *the BLS signature scheme*.

On the one hand, IIWNT11 needs to generate not only a multisignature but also a middle key, and the total data size is $Q_s + Q_v$. On the other hand, the proposed scheme does not need to generate a middle key, and the total data size is only Q_s.

Second, we compare the performance of the proposed scheme with that of IIWNT11 and show the result in Table 2. We define the following:

Table 2: Performance.

	Signing at each signer	Verification by verifier
IIWNT11	$(4w_l + 2w_d)C_a$ $+(2w_l + 1)C_b$	$(w_s + 3)C_c$
Proposed scheme	$(2w_l + w_d)C_a$ $+(w_l + 1)C_b$	$(w_s + 1)C_c$

w_l**:** The number of links.

w_d**:** The number of double-counted signers during aggregation.

w_s**:** The total number of signers.

C_a**:** The frequency of addition over an elliptic curve.

C_b**:** The frequency of the scalar product.

C_c**:** The frequency of pairing calculation.

Because both schemes generate same contents in $\mathbb{L}_i$, we exclude the calculation cost of $\mathbb{L}_i$. IIWNT11 requires the calculation cost of a middle key while signing a verification, so the performance of the proposed scheme is superior to that of IIWNT11.

However, if all documents of signers are the same, the proposed scheme cannot be used because a verifier cannot verify which paired signers are connected. In this case, IIWNT11 is used.

6 CONCLUSION

In this study, we proposed a new structured aggregate signature scheme under the GDH assumption. In addition to achieving signature authenticity and message integrity, as in ordinary signature schemes, the proposed scheme can further guarantee that the predefined signing structure has been strictly followed during signature verification. Furthermore, we showed that the proposed scheme is secure. Because of the use of the underlying GDH group, the proposed scheme has the advantages of being easy to construct and efficient with respect to key size and computational cost.

References

[1] Itakura, K. and Nakamura, K.: A public-key cryptosystem suitable for digital multisignatures: NEC Research & Development, Vol.71, pp.1–8 (1983).

[2] Burmester, M., Desmedt, Y., Doi, H., Mambo, M., Okamoto, E., Tada, M. and Yoshifuji, Y.: A structured ElGamal-type multisignature scheme: Public Key Cryptosystems - *PKC 2000*, LNCS 1751, pp.466–483, Springer-Verlag (2000).

[3] Mitomi, S. and Miyaji, A.: A Multisignature Scheme with Message Flexibility, Order Flexibility and Order Verifiability: Australasian Conference on Information Security and Privacy - *ACISP 2000*, LNCS 1841, pp.298–312, Spring-Verlag (2000).

[4] Boneh, D., Lynn, B. and Shacham, H.: Short Signatures from the Weil Pairing: Advances in Cryptology - *ASIACRYPT 2001*, LNCS 2248, pp.514–532, Springer-Verlag (2001).

[5] Lin, C. Y., Wu, T. C. and Zhang, F.: A Structured Multisignature scheme from the Gap Diffie-Hellman Group: Cryptology ePrint Archive, Report 2003/090. http://eprint.iacr.org/2003/090 (2003).

[6] Okamoto, T. and Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes: Public Key Cryptography - *PKC 2001*, LNCS 1992, pp.104–118, Springer-Verlag (2001).

[7] Joux, A. and Nguyen, K.: Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups: Cryptology ePrint Archive, Report 2001/003, http://eprint.iacr.org/2001/003 (2001).

[8] Joux, A. and Nguyen, K.: Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups: Springer J. of Cryptology, Vol.16, No.4 pp.239–247 (2003).

[9] Boneh, D., Gentry, C., Lynn, B. and Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps: Advances in Cryptology - *EUROCRYPT 2003*, LNCS 2656, pp.416–432, Springer-Verlag (2003).

[10] Inamura, M., Iwamura, K., Watanabe, R., Nishikawa, M. and Tanaka, T.: A New Tree-Structure-Specified Multisignature Scheme for a Document Circulation System: Security and Cryptography - *SECRYPT 2011*, pp.362–369, SciTePress (2011).

[11] Komano, Y., Ohta, K., Shimbo, A. and Kawamura, S.: On the security of probabilistic multisignature schemes and their optimality: In Cryptology in Malaysia - *Mycrypt 2005*, LNCS 3715, pp.132–150. Springer-Verlag (2005).

[12] Komano, Y., Ohta, K., Shimbo, A. and Kawamura, S.: Provably Secure Multisignatures in Formal Security Model and Their Optimality: IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences, Vol.E91-A, No.1, pp.107.118 (2008).

[13] Boldyreva, A.: Efficient threshold signature, multisignature and blind signature schemes based on the Gap-Diffie-Hellman-group signature scheme: Cryptology ePrint Archive, Report 2002/118, http://eprint.iacr.org/2002/118 (2002).

[14] Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme: Public Key Cryptography - *PKC 2003*, LNCS 2567, pp.31–46, Springer-Verlag (2003).

[15] Erdös, P.: Some Applications of Ramsey's Theorem to Additive Number Theory: Europ. J. Combinatorics, Vol.1, pp.43–46 (1980).