# An Order-specified Aggregate Authority-transfer Signature

Takuya Ezure and Masaki Inamura
*Graduate School of Science and Engineering, Tokyo Denki University, Saitama, Japan*

Keywords: Digital Signature, Aggregate Signature, Authority Transfer, Gap-Diffie-Hellman.

Abstract: We propose an order-specified aggregate authority-transfer signature based on the gap Diffie-Hellman group. In various organizations, to reduce the number of approvals required by someone who has relevant authority, the authority for a task can be transferred to a subordinate or another person who executes the task. Currently, authority is commonly transferred via a document, such as an authority-transfer agreement. However, to speed up the process and maintain the integrity of the evidence, we believe that it is better to transfer such authority via a computer network. In this paper, we propose an authority-transfer signature scheme using an order-specified aggregate signature and a group signature, and we propose a new authority-transfer system. In the signature method, a group signature scheme is used to express authority. Moreover, it transfers the authority owned by the manager to another member of the group. The difference from the group manager of the group signature is that this manager not only manages the group but also delegates authority. With the order-specified aggregate signature, it is possible to handle multiple signatures efficiently while verifying the order. We show that a safe and efficient authority-transfer system can be constructed using this new digital signature.

## 1 INTRODUCTION

In recent years, various systems utilizing digital signatures have been developed to improve the efficiency of the business transacted by an organization. For example, in a document-browsing system, a manager can see who has read a document and, if there have been multiple readers, the order in which the documents were read. Organizations can use this system to ensure that the workflow of approvals is done correctly. Because of the restricted validity of digital certificates, such systems are used in a closed environment such as within a single organization. However, the business of an organization is not completed only internally, and other organizations are involved. Therefore, it is necessary to make a system incorporating the employees of other organizations as well, for example, if multiple companies are jointly working on a project.

The first function required is to transfer the authority of the manager to someone else, which must be verifiable. For a general digital signature, first, a commercial digital signature is acquired from a trusted third-party certificate authority, and a manager performs key management. When the manager received electronic document which needs a signature from an employee, the manager signs it using the private key that he owns, and sends it back to the employee. However, this method relies on the manager, and if he is absent, it may take time for the digital signature to be applied. As a solution to this problem, a manager's signing authority could be delegated to an employee, for one project. Therefore, we focused on Yao et al.'s method, which can transfer authority using a group signature, which is an anonymous digital signature (Yao and Tamassia, 2009). To be precise, Yao and colleagues proposed an anonymous-signer aggregate signature scheme. However, since its function is almost the same as the group signature, it is treated as a group signature in this paper.

The second function required is the ability to aggregate multiple signatures and to reduce the size of a signature. A joint project may need multiple digital signatures. In a multisignature scheme, several people sign the same message (Itakura and Nakamura, 1983). The signatures are aggregated and rendered as a single digital signature. In an aggregate signature scheme, several people sign different messages (Boneh et al., 2003). Again, the signatures are aggregated and rendered as a single digital signature.

The third function required is to verify the order of signing. In a joint project, the order of signatures may be important in concluding contracts and the like. Therefore, we thought that it was necessary to be able



Ezure, T. and Inamura, M.
An Order-specified Aggregate Authority-transfer Signature.
DOI: 10.5220/0007359103090318
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 309-318
ISBN: 978-989-758-359-9
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

to verify the order of signing.

To summarize the above, three functions are required for a digital signature scheme for joint projects. The first is to transfer the manager's signature authority. The second is to aggregate multiple signatures to make verification more efficient. The third is that the signature order can be viewed.

In this paper, we propose an order-specified aggregate authority-transfer signature with the above three functions and an authority-transfer system that can be jointly used by multiple organizations. First, we explain Yao et al.'s group signature scheme. Next, we propose a signature scheme with a shorter verification time than the scheme of Yao et al. We propose the order-specified aggregate authority-transfer signature that adopts the structuring method of Yanai et al (Yanai et al., 2017). We use a graph for the complicated relationships between signers. Finally, we propose an authority-transfer system using the order-specified aggregate authority-transfer signature.

# 2 RELATED WORK

## 2.1 Gap Diffie-Hellman Group

Okamoto and Pointcheval defined the gap Diffie-Hellman (GDH) group (Okamoto and Pointcheval, 2001). Consider a multiplicative cyclic group $\mathbb{G}$ with prime order $p$. They defined two Diffie-Hellman (DH) problems as follows:

**Computational Diffie-Hellman (CDH) Problem:** For $a,b \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}$, given $(g, g^a, g^b)$, compute $g^{ab}$.

**Decisional Diffie-Hellman (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 $c \stackrel{?}{=} ab$.

Like the GDH group, the DDH problem is defined as easy, whereas the CDH problem is hard.

## 2.2 BLS Signature by Pairing

Various studies based on the GDH group defined in section 2.1 have been conducted. It has been shown that a GDH group on an elliptic curve can be constructed using a function called pairing. We first define the computational co-Diffie-Hellman (co-CDH) problem and the decisional co-Diffie-Hellman (co-DDH) problem on the elliptic curve. Consider different additive cyclic groups $\mathbb{G}_1'$ and $\mathbb{G}_2'$ with prime order $p$. Two problems are defined for the DH problem as follows:

**co-CDH Problem:** For $a \in \mathbb{Z}_p^*$ and $g_1 \in \mathbb{G}_1'$, $g_2 \in \mathbb{G}_2'$, given $(g_1, g_2, ag_1)$, compute $ag_2$.

**co-DDH:** For $a,b \in \mathbb{Z}_p^*$ and $g_1 \in \mathbb{G}_1'$, $g_2 \in \mathbb{G}_2'$, given $(g_1, g_2, ag_1, bg_2)$, determine whether $a \stackrel{?}{=} b$.

We will explain the features of pairing. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be different additive cyclic groups of prime order $p$ that can perform pairing. $e$ is a computable bilinear map $e$:$\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$. Pairing has the following characteristics:

- If $P_1, P_2 \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ then $e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)$.
- If $P \in \mathbb{G}_1$ and $Q_1, Q_2 \in \mathbb{G}_2$ then $e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2)$.
- If $a,b \in \mathbb{Z}_p^*$ and $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ then $e(aP, bQ) = e(bP, aQ) = e(abP, Q)$ $= e(P, abQ) = e(P, Q)^{ab}$.

When using additive cyclic groups on the elliptic curve, such as $\mathbb{G}_1$ and $\mathbb{G}_2$, it is generally considered that the co-CDH problem is hard. On the other hand, owing to the features of this pairing, the co-DDH problem can be easily solved, and the GDH group can be configured on the elliptic curve.

Boneh et al. showed that a signature scheme constructed using the features of this pairing can be realized. This is the Boneh-Lynn-Shacham (BLS) signature scheme (Boneh et al., 2001). It is based on groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and is structured as follows:

**Key Generation:** $g$ is a generator of $\mathbb{G}_1$. The private key of the signer is a random element $x \in \mathbb{Z}_p^*$, and his public key is $v = xg$.

**Signing:** $H$:$\{0,1\}^* \to \mathbb{G}_2$ is a one-way hash function. $m$ is both a plain message and a signing target. The signer computes $h = H(m)$ and returns $\sigma = xh$.

**Verification:** When the verifier is given $(g, v, m, \sigma)$, he computes $h = H(m)$ and verifies $e(g, \sigma) = e(v, h)$.

## 2.3 Aggregate Signature

An aggregate signature is a scheme in which a plurality of people sign each message, and the signatures are aggregated and represented by one digital signature. It is possible to reduce the signature size and the verification time by aggregating this plurality of signatures. On the basis of the BLS signature in section 2.2, an aggregate signature called the BGLS signature was proposed by Boneh et al., which we will explain below (Boneh et al., 2003).

Consider a group $U$ of $n$ signers who contribute to an aggregate signature. Let $i \in U$ $(1 \le i \le n)$ be a

signer participating in the aggregate signature. Each user $i$ selects a message $m_i$ to be signed and creates a signature $\sigma_i$ for it. These signatures are then combined into one BGLS signature. The security depends on the co-CDH problem. The algorithm consists of key generation, signature, aggregation, and verification. Also, like the BLS signature, we use the definition of the GDH group on the elliptic curve for $(\mathbb{G}_1, \mathbb{G}_2)$. The algorithm is structured as follows:

**Key Generation:** $g$ is a generator of $\mathbb{G}_1$. The private key of the signer $i \in U$ is a random element $x_i \in \mathbb{Z}_p^*$, and his public key is $v_i = x_i g$.

**Signing:** $H$:$\{0,1\}^* \to \mathbb{G}_2$ is a one-way hash function. Let $m_i$ be the message of signer $i$. The signer $i$ computes $h_i = H(m_i)$ and returns $\sigma_i = x_i h_i$.

**Aggregation:** Collect all individual signatures $\sigma_i$ $(1 \le i \le n)$, and calculate $\sigma = \sum_{i=1}^{n} \sigma_i$.

**Verification:** When the verifier is given $(g, v_i, m_i, \sigma)$ $(1 \le i \le n)$, he computes $h_i = H(m_i)$ $(1 \le i \le n)$ and verifies $e(g,\sigma) = \prod_{i=1}^{n} e(v_i, h_i)$.

## 2.4 Group Signature

A group signature is a digital signature scheme in which a verifier can confirm only the affiliation of a signer to a group and cannot identify who signed it. The group signature can protect privacy by preventing identification of the signer.

This section describes the aggregate group signature proposed by Yao et al (Yao and Tamassia, 2009). For simplicity, some methods have been changed. This signature scheme was created based on the group signature proposed by Chen et al (Chen et al., 2006). This group signature can be made using the BGLS signature algorithm. Therefore, because the security of the BGLS signatures has been proved, we can prove the security of the aggregate group signatures.

Here, we explain how a group signature is created from a BGLS signature. In advance, members create secret keys and public keys, and send the public key to group manager GM. The GM performs the BGLS signature (group signature key) on the public key and returns it to the member. The group signature is the aggregate of the BGLS signature of the message and the group signature key.

As a result, not only the group signatures of the same group but also the group signatures of other groups can be aggregated together. In addition, information such as an expiry date can be included in the group signature key of each member issued by the GM, and this information can be verified. Further, Yao and colleagues proposed a method of delegating authority by putting authority information into a group signature key.

An aggregate group signature is organized as follows:

**Preconditions:**

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are the additive cyclic group and the multiplicative cyclic group of the prime order $q$, respectively, where $\mathbb{G}_1$ is the GDH group.
- $\pi$ is the generator of $\mathbb{G}_1$.
- A map $e$:$\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map.
- $H$:$\{0,1\}^* \to \mathbb{G}_1$ is a one-way hash function.

**Set-up:**

- A trusted third-party creates the parameter para$(\mathbb{G}_1, \mathbb{G}_2, e, \pi, H)$.
- Member $U$ chooses a private $s_u \in \mathbb{Z}_q^*$ as his private key and computes $P_u = s_u \pi$ as his public key.
- Similarly, the GM chooses a private key $s_A \in \mathbb{Z}_q^*$ and computes the public key $P_A = s_A \pi$.

**Join:**

- $U$ randomly chooses a number of privates $(x_1, \ldots, x_l \in \mathbb{Z}_q^*)$ and computes one-time signing factors $X_{u,1} = x_1\pi, \ldots, X_{u,l} = x_l\pi$ and one-time signing public keys $K_{u,1} = s_u x_1 \pi, \ldots, K_{u,l} = s_u x_l \pi$. Keys $P_u$, $X_{u,i}$, and $K_{u,i}$ are sent to the GM, for all $i \in [1,l]$.
- The GM tests if $e(K_{u,i}, \pi) = e(P_u, X_{u,i})$ for all $i$. If the test fails, the protocol terminates. Otherwise, the GM runs the BGLS signing algorithm on inputs $s_A$ and strings $T \| K_{u,i}$ ($T$ is the authority information) to obtain $S_{u,i} = s_A H(T \| K_{u,i})$ for all $i$. $S_{u,i}$ is the ith one-time signing permit for $U$ and is given to $U$. The GM adds tuples $(P_u, K_{u,i}, X_{u,i})$ to his record for all $i$.

**Sign:**

- $U$ runs the BGLS signing algorithm with private key $s_u x_i$ and message $M$ and obtains a signature $S_u = s_u x_i H(M)$.
- $U$ aggregates signature $S_u$ with one-time signing permit $S_{u,i}$ associated with private $s_u x_i$. This is done by running the aggregation function of the BGLS scheme, which returns a signature $S_g = S_u + S_{u,i}$.

**Aggregate:**

This is the same as the aggregation algorithm in the BGLS scheme. It takes as inputs n signatures $S_{g_k}$ and the corresponding values $P_{A_k}$ and $K_{u_k,i_k}$ for all $k \in [1,n]$. Set $S_{Agg} = \sum_{k=1}^{n} S_{g_k}$.
$S_{Agg}$ is outputted as the aggregate group signature. Note that the $k$th GM public key $P_{A_k}$ for $k \in [1,n]$

does not need to be the same. In other words, signatures from different organizations can be aggregated.

**Verify:**

- For $1 \leq k \leq n$, compute the hash digest $H(M_k)$ of message $M_k$ and $h_k = H(T_k \| K_{u_k,i_k})$ of the statement on the one-time signing permit.
- $S_{Agg}$ is accepted if $e(S_{Agg}, \pi) = \prod_{k=1}^{n} e(H(M_k), K_{u_k,i_k}) e(h_k, P_{A_k})$.

**Open:**

If $S_{Agg}$ is valid, the GM can easily identify a member's public key $P_{u_k}$ from $K_{u_k,i_k}$ by consulting the record.

# 3 AGGREGATE AUTHORITY-TRANSFER SIGNATURES

In this paper, we assumed that a plurality of people, including those with different memberships, can verifiably sign a document. We may want to verify a signer's identity, their affiliation, or authority at the time of signing. We can do this using Yao et al.'s method, as discussed in section 2.4 (Yao and Tamassia, 2009). We believe that strong anonymity of a group signature is not necessary. Because an aggregate transfer signature is not a group signature. Moreover, as the number of signatures increases, the verification time increases. In this section, we propose an aggregate transfer signature with short verification time based on Yao et al.'s method. Unlike group manager GM, the manager M of this signature scheme not only manages groups but also delegates authority.

## 3.1 Security Definition

In this section, we describe the security of an aggregate authority-transfer signature. A signature is defined as secure when it satisfies the following: correctness, unforgeability, and traceability. It is the same as the security of a group signature without anonymity and unlinkability.

**Correctness:** A signature created by a legitimate member passes verification and identifies the signer.

**Unforgeability:** Only legitimate members can make valid signatures. Even if an attacker collaborates with another member or manager, it can not forge the signature of a member who is not collusion.

**Traceability:** The signature that passes the verification can always identify the signer. Even if an attacker collaborates with other members, it is impossible to create a signature that the verification passes but the signer cannot identify.

## 3.2 Construction

**Preconditions:**

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are the additive cyclic groups of the prime order $q$.
- $\mathbb{G}_1$ and $\mathbb{G}_2$ are the co-GDH groups.
- $\mathbb{G}_3$ is the multiplicative cyclic group of prime order $q$.
- $P$ is is the generator of $\mathbb{G}_1$.
- $Q$ is is the generator of $\mathbb{G}_2$.
- A map $e$:$\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ is a bilinear map.
- $H_1$:$\{0,1\}^* \times \mathbb{G}_2 \to \mathbb{G}_1$ and $H_2$:$\{0,1\}^* \times \mathbb{G}_2 \to \mathbb{Z}_q^*$ are one-way hash functions.

**Set-up:**

- A trusted third-party creates the parameter para$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, P, Q, e, H_1, H_2)$.
- Member $U$ chooses a private $s_u \in \mathbb{Z}_q^*$ as his private key and computes $P_u = s_u P$ as his public key.
- Similarly, the manager M chooses his private key $s_A \in \mathbb{Z}_q^*$ and computes the public key $P_A = s_A Q$.

**Join:**

- $U$ randomly chooses a private $x \in \mathbb{Z}_q^*$ and computes signing factors $X_u = xQ$ and public keys $K_u = xs_u Q$. Keys $P_u$, $X_u$, and $K_u$ are sent to the M.
- The M confirms that $P_u$ is $U$'s public key. The GM tests if $e(P, K_u) = e(P_u, X_u)$. If the test fails, the protocol terminates. Otherwise, the M runs the BGLS signing algorithm on inputs $s_A$ and strings $T_u \| K_u$ ($T_u$ is the authority information transferred by manager) to obtain $S_u = s_A H_1(T_u \| K_u)$. $S_u$ is an authority-transfer key and is given to $U$. The M adds tuples $(P_u, K_u, X_u, T_u)$ to his record.

**Sign:**

$U$ signs message M. $U$ gets an authority-transfer key $S_u$ from the M. $U$ randomly chooses $r \in \mathbb{Z}_q^*$. $U$ computes the following:

$$B = rQ. \tag{1}$$

$$C = S_u + H_2(M \| B) x s_u P + rP. \tag{2}$$

$$K_u = xs_u Q. \tag{3}$$

The signature is $\sigma = \{B, C, K_u, T_u\}$.

**Aggregate:**

Aggregation takes as inputs n signatures, $\sigma_k = \{B_k, C_k, K_{u_k}, T_{u_k}\}$, and the corresponding values $P_{A_k}$ and $K_{u_k}$ for all $k \in [1,n]$. Only $C_k$ can be aggregated and becomes $C_{Agg} = \sum_{k=1}^{n} C_k$. The aggregate authority-transfer signature is $\sigma_{Agg} = \{B_k, C_{Agg}, K_{u_k}, T_{u_k}\}$ for all $k \in [1,n]$.

**Verify:**

- For $1 \leq k \leq n$, compute the hash digests $H_1(K_{u_k} \| T_{u_k})$ and $H_2(M_k \| B_k)$.
- $\sigma_{Agg}$ is accepted if $e(C_{Agg}, Q) = \prod_{k=1}^{n}(e(H_1(K_{u_k} \| T_{u_k}), P_{A_k})) \cdot e(P, \sum_{k=1}^{n}(H_2(M_k \| B_k)K_{u_k} + B_k))$.

**Open:**

If $\sigma_{Agg}$ is valid, the M can easily identify a member's public key $P_{u_k}$ from $K_{u_k}$ by consulting the record.

## 3.3 Construction

In this section, we describe the advantages and disadvantages of the aggregate authority-transfer signature compared with Yao et al.'s method (Yao and Tamassia, 2009). The following are the verification formulas of Yao et al.'s method and the aggregate transfer signature when N signatures are aggregated.

**The Verification Formula for Yao et al.'s Method:**

$$e(S_{Agg}, \pi) = \prod_{k=1}^{n} e(H(M_k), K_{u_k,i_k}) e(h_k, P_{A_k}). \quad (4)$$

**The Verification Formula for Aggregate Authority-transfer Signature:**

$$e(C_{Agg}, Q) = \prod_{k=1}^{n}(e(H_1(K_{u_k} \| T_{u_k}), P_{A_k})) \cdot e(P, \sum_{k=1}^{n}(H_2(M_k \| B_k)K_{u_k} + B_k)). \quad (5)$$

The number of pairings calculated when verifying the aggregate signature for $N$ signatures for each formula is shown in table 1

From table 1, it can be seen that the number of pairings at the time of verification of the aggregate authority-transfer signature is about half of that of Yao et al.'s method if the number of signatures, $N$, is large. An advantage is that the verification time can be shortened by reducing the number of pairings. A disadvantage is that the number of signatures of a point on the elliptic curve increases by 1 for each message. However, since the points on the elliptic curve are not large, the advantage of shortening the verification time is greater than the disadvantage that the signature size increases.

Table 1: Number of pairings when verifying an aggregate signature ($N$ signatures).

| Yao et al.'s method | Aggregate authority-transfer signature |
|---|---|
| $2N+1$ | $N+2$ |

## 3.4 Security Analysis

This section explains the security of the aggregate authority-transfer signature described in section 3.2. First, we will prove the security of a single authority-transfer signature that does not aggregate. Next, we prove the security of the aggregate authority-transfer signature.

### 3.4.1 Correctness

The correctness of the authority-transfer signature is proved as follows:

$$\begin{aligned} & e(C, Q) \\ =& e(S_u + H_2(M\|B)xs_uP + rP, Q) \\ =& e(S_u, Q) \cdot e(H_2(M\|B)xs_uP, Q) \cdot e(rP, Q) \\ =& e(s_A H_1(T_u\|K_u), Q) \cdot e(H_2(M\|B)xs_uP, Q) \\ & \cdot e(rP, Q) \\ =& e(H_1(T_u\|K_u), s_A Q) \cdot e(P, H_2(M\|B)xs_uQ) \\ & \cdot e(P, rQ) \\ =& e(H_1(T_u\|K_u), P_A) \cdot e(P, H_2(M\|B)K_u) \\ & \cdot e(P, B) \\ =& e(H_1(T_u\|K_u), P_A) \cdot e(P, H_2(M\|B)K_u + B). \end{aligned} \quad (6)$$

### 3.4.2 Unforgeability

In the authority-transfer signature scheme, attackers can be either a member or a manager. We do not consider cases where the third party is an attacker because fewer information is available than other attackers.

**The ability to attack when the attacker is a member of the group:**

- Create and register new members.
- Issue authority-transfer authority key $A$ to a new member.
- Collusion between a new member and another member (non-collusion with a manager).

**The attack target when the attacker is a member of the group:**
Passing verification using $K$ for a member other than the member who has colluded or the new member.

**The ability to attack when the attacker is a manager:**

- Create and register new members.
- Issue authority-transfer authority key $A$ to a new member.
- Collusion between a new member and an existing member.

**The attack target when the attacker is a manager:**
Passing verification using $K$ for members other than the member who has colluded or the new member.

Security is considered by looking at the signature formula. The signature $C = S_u + H_2(M\|B)xs_uP + rP$ can be divided into the following:

$$A = S_u = s_A H_1(T_u\|K_u). \quad (7)$$
$$D = H_2(M\|B)xs_iP + rP. \quad (8)$$

The respective verification equations are the following:

$$\begin{aligned} e(A,Q) &= e(s_A H_1(T_u\|K_u), Q) \\ &= e(H_1(T_u\|K_u), s_A Q). \end{aligned} \quad (9)$$
$$\begin{aligned} e(D,Q) &= e(H_2(M\|B)xs_iP + rP, Q) \\ &= e(P, H_2(M\|B)K_u)\cdot e(P,B). \end{aligned} \quad (10)$$

The authority-transfer key $A$ of formula (7) and signatures $K$ and $T$ are keys issued during the interaction between the member and a manager. Therefore, there is a possibility that the key information is leaked because of communication leakage or collusion. In addition, old signatures can also be obtained. We set the following attack environment which is favorable to attackers.

**Attacker's Environment:** An attacker can obtain all authority-transfer keys $A$, signatures $\sigma = \{B, C, K_u, T_u\}$, and messages $M$ issued, including the forgery target.

For an attacker to forge a signature, they must forge both expression (9) and expression (10). Expression (9) is the same as the BLS signature, where the message is the authority information $T_u$ and the signature $K_u$. Forging the authority information $T_u$ and the signature $K_u$ corresponding to the message is difficult because of the security of the BLS signature. It is difficult for an attacker who is a member to forge a signature other than the signature $K_u$. Therefore, the attack target of an attacker who is a member is the same as the attack target of an attacker who is a manager. The attacker tries to forge expression (10) using the signature $K_u$, where the verification formula of expression (9) is established. There are roughly two methods for forging expression (10).

- Obtain $xs_u$ from the signature $K_u = xs_uQ$ made by the forgery target. Alternatively, obtain $r_n$ from signature $B = rQ$.

This is difficult because it is a discrete logarithm problem on an elliptic curve.

- Obtain $H_2(M\|B)xs_uP$ or $rP$ from the signature $C$ made by the forgery target. Then, extract $H_2(M\|B)$ from $H_2(M\|B)xs_uP$, find $xs_uP$, and use it for forgery.

If it is possible to find the same signature $B = rQ$ in multiple signatures, there is a possibility that $xs_uP$ can be obtained, but since the $r$ used for a signature is random, the probability of a match is low.

#### 3.4.3 Traceability

Expression (9) of the verification expression is a BLS signature in which the message is the authority information $T_u$ and the signature $K_u$. Therefore, it is difficult to make a signature that can pass verification with the signature $K_u$ that cannot be tracked because of the security of the BLS signature, even if a member collaborates with another member.

#### 3.4.4 Unforgeability of the Aggregate Authority-transfer Signature

The security of the aggregate authority transfer signature will be briefly described. Authority transfer signature and aggregate authority transfer signature indicate that unforgeability is equivalent.

We prove this security using the method of Boldyreva (Boldyreva, 2002; Boldyreva, 2003), which is used in Inamura et al.'s system as the security proof (Inamura et al., 2011; Inamura and Iwamura, 2013). Let $A$ be an attacker who is attempting to forge an aggregate authority-transfer signature $C'_{Agg}$. Let $B$ be an attacker who forges an authority-transfer signature. If $A$'s attack succeeds, it indicates that $B$'s attack succeeds. If $B$'s attack succeeds, it is obvious that $A$'s attack will succeed, so the proof of this is omitted.

Attacker $B$ has a verification key of the object to be forged $\sigma_1 = \{K_{u_1}, T_{u_1}, P_{A_1}, B_1\}$ and responds to the random oracle and the signature oracle. Attacker $B$ executes $A$ as an honest player. First, $B$ gives $\sigma_1 = \{K_{u_1}, T_{u_1}, P_{A_1}, B_1\}$ to $A$. Attacker $A$ outputs other signature keys and verification keys

$$\left\{ \begin{array}{c} (S_{u_2}, x_2 s_{u_2}, r_2, T_{u_2}, K_{u_2}, B_2, P_{A_2}), \ldots, \\ (S_{u_n}, x_n s_{u_n}, r_n, T_{u_n}, K_{u_n}, B_n, P_{A_n}) \end{array} \right\}.$$

Also, attacker $A$ uses the random oracle and the signature oracle to find $C'_{Agg}$ and the message $(M_1, \ldots, M_n)$ to be signed, and responds to $B$. Attacker $B$ computes the following using $C'_{Agg}$:

$$C'_{Agg} - \sum_{i=2}^{n} (S_{u_i} + H_2(M_i \| B_i) x_i s_{u_i} P + r_i P)$$
$$= S_{u_1} + H_2(M_1 \| B_1) x_1 s_{u_1} P + r_1 P.$$

Therefore, the authority-transfer signature corresponding to $\sigma_1$ can be computed, and $B$'s attack succeeds.

As described above, it is difficult to forge the aggregate authority-transfer signature if forgery of the authority-transfer signature is difficult.

# 4 ORDER-SPECIFIED AGGREGATE AUTHORITY-TRANSFER SIGNATURE SCHEME

In this proposed method, we adopt the structuring method proposed by Yanai et al. for the aggregate authority-transfer signature described in section 3 (Yanai et al., 2017). Using graphs, we show the complex relationships between signers. As a method of structuring, when signing, we refer to the graph. The messages of all of the signers before the current signer are gathered and signed in addition to the current message. The generated signature and the signatures for the adjacent signers are aggregated, and a new aggregate signature is created. For a proof of security, see Yanai et al.

## 4.1 Series-parallel Graph

### 4.1.1 Definition of a Series-parallel Graph

Let $G$ be a set of graphs. A series-parallel graph is a graph generated by applying either a serial graph or a parallel graph recursively in an arbitrary order. 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 following step 1 or step 2.

1. With a unique label $i$ in $G$, $G_i(I_i,T_i)$ is composed of one edge connecting $I_i$ and $T_i$. We call such a graph an atomic graph and denote it by $\phi_i \in G$.
2. For step 2, either one of the following steps is executed:
   (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 $T = T_1 = T_2 = \cdots = T_n$.
   (Serial graph) Given $n$ graphs $G_i(I_i,T_i)$, for $1 \leq i \leq n$, construct $G(I,T)$ by setting $I = I_1$, $T_1 = I_2, \ldots, T_{n-1} = I_n$, and $T_n = T$.

Intuitively, in the above definitions, constructing $G(I,T)$ means compositions of $n$ atomic graphs $\phi_i \in G$ for $i = [1,n]$ either as a serial one or a parallel one (Yanai et al., 2017).

### 4.1.2 Graph Composition

For two atomic graphs $\phi_1, \phi_2 \in G$, 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 $T = T_1 = T_2$, 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 $T(i)$ a set of graphs connecting to the initial vertex $I_i$ of the ith graph in a way such that
$T(i) = \{x | I_i = T_x \wedge 1 \leq x < i \wedge G_x(I_x,T_x) \subset \psi_n\}$,
where n is the number of atomic graphs and $\psi_n$ is the whole graph for any $n$, by $I(i)$ a set of graphs connecting to the terminal vertex $T_i$ of the $i$th graph in a way such that
$I(i) = \{x | T_i = I_x \wedge i < x \leq n \wedge G_x(I_x,T_x) \subset \psi_n\}$,
by $\{a_j\}_{j \in T(i)}$, for all $a_j$ for $j \in T(i)$. A whole graph that includes multiple graphs in its terminal vertex $T_i$ can be denoted by $\psi_{I(i)}$. That is, if $\psi_{I(i)}$ includes a single atomic graph $\phi_i$ in the terminal vertex, then $\psi_{I(i)}$ is equal to $\psi_i$ (Yanai et al., 2017).

### 4.1.3 Weight of a Graph

We define a weight function $\omega_i(\psi_n)$ that represents a weight of each label $i$ for a graph $\psi_n$. Intuitively, $\omega_i(\psi_n)$ is the number of paths including an edge with a label $i$ from $I_i$ to $T_n$ for $\psi_n$. We denote by $\#\psi_n$ the number of edges in $\psi_n$ for any structure $\psi_n$ (Yanai et al., 2017).

### 4.1.4 Toy Example

In this section, we show a toy example of a series-parallel graph and its parameters. For a graph shown in figure 1, its parameters are given as shown in table 2.

We call the initial node for the whole graph as "whole initial" and the terminal node for the whole graph as "whole terminal", respectively. In the columns of $T(i)$ and $I(i)$, we give the indexes of cor-

Table 2: Signer structure.

| $\phi_i$ | Graph composition in $\psi_i$ | $T(i)$ | $I(i)$ | $\omega_i(\psi_n)$ |
|---|---|---|---|---|
| $\phi_1$ | $\psi_1 = \phi_1$ | Whole initial | 2,3 | 3 |
| $\phi_2$ | $\psi_2 = \phi_1 \cap \phi_2$ | 1 | Whole terminal | 1 |
| $\phi_3$ | $\psi_3 = \phi_1 \cap \phi_3$ | 1 | 4,5 | 2 |
| $\phi_4$ | $\psi_4 = \phi_1 \cap \phi_3 \cap \phi_4$ | 3 | Whole terminal | 1 |
| $\phi_5$ | $\psi_5 = \phi_1 \cap \phi_3 \cap \phi_5$ | 3 | Whole terminal | 1 |

responding atomic graphs. We also denote by $\psi_n$ as whole graph.

We show a toy example of the extraction in table 3.

We utilize a series-parallel graph as the 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 that represents the position of each signer in any structure (Yanai et al., 2017).

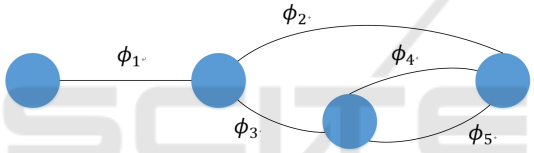


Figure 1: Example of graph $\psi_n$.

## 4.2 Construction

**Preconditions:**

The preconditions are the same as those for the aggregate authority-transfer signature in section 3.2.

**Set-up:**

The setup is the same as that for the aggregate authority-transfer signature in section 3.2.

**Join:**

The join condition is the same as that for the aggregate authority-transfer signature in section 3.2.

**Sign:**

$U$ is the $i$th signer and signs message $M_i$ for $\psi_i$. $U$ refers to the graph before the signature, and it verifies the signature of the previous signer. We pass $(\text{para}, \{P_{A_j}, M_j\}_{j\subset\psi_{T(i)}}, \psi_{T(i)}, \{\sigma_{Agg_j}\}_{j\in T(i)})$ to the verification algorithm. If the verification fails, the protocol terminates. Otherwise, $U$ randomly chooses $r \in \mathbb{Z}_q^*$. $U$ computes the following:

$$B_i = rQ. \tag{11}$$
$$K_{u_i} = xs_uQ. \tag{12}$$
$$h_i = H_2(M_i \| B_i). \tag{13}$$
$$\{h_j = H_2(M_j \| B_j)\}_{j\subset\psi_{T(i)}}. \tag{14}$$

Finally, $U$ computes
$C_i = \sum_{j\in T(i)} C_j + S_{u_i} + xs_u(\sum_{j\subset\psi_{T(i)}} h_j + h_i)P + rP$.
$S_{u_i} = s_{A_i} H_1(T_{u_i} \| K_{u_i})$ is an authority-transfer key. $T_{u_i}$ is the authority information transferred by the M. The order-specified aggregate authority-transfer signature is
$\sigma_{Agg_i} = (\{B_j, K_{u_j}, T_{u_j}\}_{j\subset\psi_i}, C_i)$.

**Verify:**

Given $(\text{para}, \{P_{A_j}, M_j\}_{j\subset\psi_{I(i)}}, \psi_{I(i)}, \{\sigma_{Agg_j}\}_{j\in I(i)})$, we check if $i \le 3\frac{\log P}{\log 3}$ holds (Tada, 2003). If not, the protocol terminates. Next, we check whether all of the authority information $\{T_{u_j}\}_{j\subset\psi_{I(i)}}$ is appropriate. The verifier computes $C = \sum_{j\in I(i)} C_j$ if $|\{\sigma_{Agg_j}\}_{j\in I(i)}| > 1$. Otherwise, if $|\{\sigma_{Agg_j}\}_{j\in I(i)}\}| = 1$, we set $C = C_j$. Then, we check if the following equation holds with $H_1(K_{u_j} \| T_{u_j})$ and $H_2(M_j \| B_j)$ for all $j$:

$$\begin{aligned} & e(C, Q) \\ &= \prod_{j\subset\psi_{I(i)}} e\Big(\omega_j(\psi_{I(i)}) H_1(K_{u_j} \| T_{u_j}), P_{A_j}\Big) \\ &\quad \cdot e\Bigg(P, \sum_{j\subset\psi_{I(i)}} \Bigg( \Big( \sum_{l\subset\psi_{T(j)}} (H_2(M_l \| B_l)) \\ &\qquad + H_2(M_j \| B_j)\Big) \omega_j(\psi_{I(i)}) K_{u_j} \Bigg) \\ &\qquad + \sum_{j\subset\psi_{I(i)}} \omega_j(\psi_{I(i)}) B_j \Bigg). \end{aligned}$$

**Open:**

Open is the same as that for the aggregate authority-transfer signature in section 3.2.

Table 3: Signer structure.

| $\psi_i$ | Indexes $j \in \psi_i$ |
|---|---|
| $\psi_1$ | 1 |
| $\psi_2$ | 1,2 |
| $\psi_3$ | 1,3 |
| $\psi_4$ | 1,3,4 |
| $\psi_5$ | 1,3,5 |

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

# 5 PROPOSED AUTHORITY-TRANSFER SYSTEM

In this section, we propose an authority-transfer system using the order-specified aggregate authority-transfer signature proposed in section 4. This system can be used by multiple organizations. This system has the following features:

- The number of digital certificates issued by the trusted authority is lower. The number of certificates does not depend on the number of members.
- Managers can easily transfer their authority to members.
- The authority of the signer can be verified.
- Signatures of different organizations can be aggregated.
- The verification time is small.
- The order of signing can be verified.

## 5.1 Entities

In the authority-transfer system, there are five entities (figure 2). We will explain the role of each.

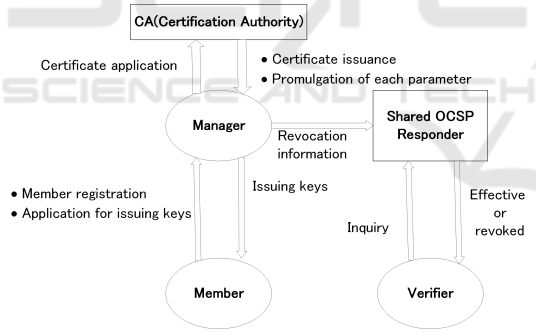


Figure 2: Configuration of the authority-transfer system.

### 5.1.1 Certificate Authority

A certificate authority is a trusted third-party. It issues commercial digital certificates based on an application from a manager in an organization. All organizations participating in this system must use certificates from the same certificate authority so that the same parameters are used for signing and verification.

### 5.1.2 Member

Members are employees of an organization. They register in the group and apply for a manager to issue an authority-transfer key.

### 5.1.3 Online Certificate Status Protocol Responder

The online certificate status protocol (OCSP) responder holds the revocation information for a signature and can respond in real time to inquiries about the validity of the signature. If each organization has an OCSP responder, the verifier needs to communicate with each of them, which takes time. Therefore, all organizations participating in this system use the same OCSP responder, which reduces the verification time.

### 5.1.4 Manager

Each organization has a manager, who manages the commercial digital signatures authenticated by the certificate authority. When a manager receives an application for a key from an employee in his organization, he registers the employee with the group. Then, he issues an authority-transfer key and transfers his authority to sign to the member, thus delegating his authority.

### 5.1.5 Verifier

The verifier verifies the order-specified aggregate authority-transfer signature. The verifier queries the OCSP responder whether the signer has not been revoked.

## 5.2 Construction

**Preconditions:**

The preconditions are the same as those for the aggregate authority-transfer signature in section 3.2.

**Set-up:**

- A trusted third-party certificate authority creates parameters $\mathrm{para}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, P, Q, e, H_1, H_2)$.
- Member $U$ chooses a private $s_u \in \mathbb{Z}_q^*$ as his private key and computes $P_u = s_u P$ as his public key.
- Similarly, the manager M chooses his private key $s_A \in \mathbb{Z}_q^*$ and computes the public key $P_A = s_A Q$.
- The M sends the public key $P_A$ to the certificate authority to apply for a certificate.
- The certificate authority reviews the application and sends a certificate to the M.

**Join:**

This is almost the same as the join condition for

the aggregate authority-transfer signature in section 3.2. Also included in the authority information $T$ is the serial number for the revocation information.

**Sign:**
This is the same as that for the order-specified aggregate authority-transfer signature in section 4.2.

**Verify:**
A revocation check is added to the verification for the order-specified aggregate authority-transfer signature in section 4.2:

- The verifier transmits the serial number included in the authority information $T$ of all signatures to be verified to the OCSP responder.
- The OCSP responder refers to the revocation list and checks for the serial number from the verifier. If it is not in the list, he sends a "valid" message to the verifier; otherwise, he sends an "invalid" message.
- If the verifier receives an "invalid" message from the OCSP responder, the verification fails.

**Open:**
This is the same as that for the aggregate authority-transfer signature in section 3.2.

**Management of Revocation Information:**
If a manager wants to invalidate a key, he sends the serial number in the authority information $T$ to the OCSP responder.

# 6 CONCLUSIONS

In this paper, we proposed an order-specified aggregate authority-transfer signature that can be used to delegate authority, aggregate signatures, and maintain the order of signing. Moreover, we proposed an authority-transfer system using that signature scheme.

An advantage of an aggregate authority-transfer signature is that the verification time is reduced compared to that in Yao et al.'s method. A disadvantage is that signature size is larger compared to that in Yao et al.'s method. For this reason, we believe that the proposed method should be operated in a system that requires a quick response.

As a future work, we will consider the security of the order-specified aggregate authority-transfer signatures, implement the authority-transfer system, and consider whether the method is practical.

# ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Number JP16K00192.

We would like to thank Enago for the English language review (https://www.enago.jp).

# REFERENCES

Boldyreva, A. (2002). Efficient threshold signature, multisignature and blind signature schemes based on the gap-diffie-hellman-group signature scheme. *Cryptology ePrint Archive*, Report 2002/118.

Boldyreva, A. (2003). Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography - PKC 2003, LNCS*, volume 2567, pages 31–46. Springer.

Boneh, D., Gentry, C., Lynn, B., and Shacham, H. (2003). Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - EUROCRYPT 2003, LNCS*, volume 2656, pages 416–432. Springer.

Boneh, D., Lynn, B., and Shacham, H. (2001). Short signatures from the weil pairing. In *Advances in Cryptology - ASIACRYPT 2001, LNCS*, volume 2248, pages 514–532. Springer.

Chen, X., Zhang, F., and Kim, K. (2006). New id-based group signature from pairings. *Science Press J. of Electronics (China)*, 23(6):892–900.

Inamura, M. and Iwamura, K. (2013). Content approval systems with expansions of a new pair-connected-structured aggregate signature scheme. *IGI Global International J. of E-Entrepreneurship and Innovation*, 4(2):15–37.

Inamura, M., Iwamura, K., Watanabe, R., Nishikawa, M., and Tanaka, T. (2011). A new tree-structure-specified multisignature scheme for a document circulation system. In *International Conference on Security and Cryptography - SECRYPT 2011*, pages 362–369. SciTePress.

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

Okamoto, T. and Pointcheval, D. (2001). The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography - PKC 2001, LNCS*, volume 1992, pages 104–118. Springer.

Tada, M. (2003). A secure multisignature scheme with signing order verifiability. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(1):73–88.

Yanai, N., Iwasaki, T., Inamura, M., and Iwamura, K. (2017). Provably secure structured signature schemes with tighter reductions. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E100-A(9):1870–1881.

Yao, D. and Tamassia, R. (2009). Compact and anonymous role-based authorization chain. *ACM Trans. on Information and System Security*, 12(3):15:1–15:27.