ID-based aggregate proxy signature scheme
realizing warrant-based delegation*

YEN-CHING LIN2, TZONG-CHEN WU1,2, and JIA-LUN TSAI2
1Taiwan Information Security Center at National Taiwan University of Science and Technology (TWISC@NTUST), Taipei 106, Taiwan
2Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan

This paper proposes a novel ID-based aggregate proxy signature scheme that realizes a warrant-based delegation for an original signer to transfer his/her signing power to a given set of proxy signers. Our proposed scheme allows $n$ distinct proxy signers to sign $n$ distinct messages in such a way that these $n$ individual signatures can be aggregated into a single one without expansion. In the practical applications, such specific kind of aggregate signatures is significantly applausive for enforcing the delegation of authority with both bandwidth and computation savings. Our proposed scheme requires constant bilinear pairing operations for signature verification. Besides, the size of the aggregate proxy signature is the same as that of each of the individual proxy signatures, regardless of the number of participant proxy signers has involved. We also formally sketch the security model of our proposed scheme and show that it is secure against the chosen message attacks under the computational Diffie-Hellman (CDH) assumption.

Keywords: ID-based signature, aggregate proxy signature, bilinear pairing, chosen message attack, computational Diffie-Hellman assumption

1. INTRODUCTION

In 2003, Boneh et al. [1] introduced the concept of aggregate signature that allows $n$ distinct signers to sign $n$ distinct messages in such a way that these $n$ individual signatures can be aggregated into a single one without expansion. Since then, several aggregate signature schemes or their variants have been developed for achieving this purpose [2-7]. The aggregate signature scheme is very useful in real-world applications. For example, in the scenario of the secure Border Gateway Protocol (BGP) as described in [8], each router successively signs its own segment of a path in the network, and then forwards the collection of signatures associated with the path to the next router. The aggregate signature scheme can be used to compress these signatures into a single one, and hence reduce the overheads of both bandwidth and computation required in the original secure BGP.

Extended from the spirit of the aggregate signature scheme addressed in [1], an aggregate proxy signature scheme allows an original signer to delegate his/her signing authority to a set of proxy signers and distinct proxy signers can sign distinct messages, respectively, in such a way that these individual proxy signatures can be further

*This paper was partially supported by National Science Council, ROC, under the grant numbers NSC 98-2221-E-011-073-MY3, NSC 99-2218-E-011-011, and NSC 100-2219-E-011-002.
aggregated into a single one. The aggregate proxy signature scheme is plausible for enforcing the delegation problem in some commercial applications. For example, the chief executive of the company, in case of absence, can delegate his/her signing power to some department directors for signing distinct parts of a contract in accordance with their expertise by following a given warrant. Note that the aggregate proxy signature scheme is somewhat different from the previously proposed multi-proxy signature schemes, such as those in [9-11]. In those previous works, each proxy signer will sign the same message with or without following a predefined signing order, although they preserve the feature that the resulting multi-proxy signature has the same size as that of each individual proxy signature.

In this paper, we shall propose an ID-based aggregate proxy signature scheme that can realize the warrant-based delegation stated above. Our proposed scheme only requires constant bilinear pairing operations for signature verification. Besides, the size of the aggregate proxy signature is the same as that of each individual proxy signature, regardless of the number of participant proxy signers has involved. We also formally sketch the security model of the aggregate proxy signatures and show that our proposed scheme is secure against the chosen message attacks under the computational Diffie-Hellman (CDH) assumption [12].

2. OUR PROPOSED SCHEME

The participant roles involved in the system include the private key generator \( PKG \), the original signer \( U_0 \), the set of \( n \) proxy signers \( \{ U_1, U_2, \ldots, U_n \} \), the aggregator \( AGG \), and the verifier \( VER \). Our proposed scheme consists of the following phases: System Setup, Key Generation, Delegation, Proxy Signature Generation, Aggregation, and Aggregate proxy signature Verification. Functional specifications of these phases are briefly described as below.

Like the system model proposed in [3-7, 9-11, 13-14], \( PKG \) is assumed to be a Trusted Third Party who is responsible for system setup as well as generating a private-public key pair for himself/herself. After system setup, \( PKG \) uses his/her own private key as the seed to generate the private keys for the original signer and the proxy signers associated to their identities \( ID_i \) that are used as the public keys, respectively. The original signer’s private key \( S_0 \) is used for signing the warrant be made, and the proxy signer’s private key \( S_i \) is used for signing the message in accordance with the signing power delegated from the original signer specified in the warrant. When \( U_0 \) wants to delegate his/her signing power to the proxy signers \( U_1, U_2, \ldots, U_n \), he/she first makes a warrant \( w \) and its corresponding signature \( \sigma_0 \) for this delegation. The warrant \( w \) contains the information regarding the identity of original signer, the identities of the proxy signers, and the start-time and the end-time period of the delegation, as specified in [15]. Any proxy signer \( U_i \) can further confirm the delegation by verifying the signature \( \sigma_0 \) of the warrant \( w \) and only can sign the messages during the valid period of the delegation. Under the delegation specified in the warrant \( w \), the proxy signer \( U_i \) can generate a proxy signature \( \sigma_i \) for the message \( m_i \) at the time \( T_i \), and then sends \( \{ ID_i, m_i, T_i, \sigma_i \} \) to \( AGG \) for further aggregation. For simplicity, any participant proxy signer can serve as \( AGG \) in practice. The aggregate proxy signature \( \sigma_{AM} \) for the messages \( AM = \{ m_1, m_2, \ldots, m_n \} \) with the identities \( AID = \{ ID_1, ID_2, \ldots, ID_n \} \) at the individual signing time \( AIT = \{ T_1, \ldots, T_n \} \),
$T_2, \ldots, T_n$ is directly constructed from all $\sigma_i$'s under the given $w$. Anyone with only knowing the public key of $PKG$ and the identities of the original signer and the participant proxy signers can serve as $VER$ to verify the validity of $\sigma_{AM}$ without regarding to the signing order among the participant proxy signers. The system model of our proposed scheme is shown in Figure 1. Details of these phases are stated in the following.

**System Setup.** Initially, $PKG$ defines the following system parameters:
- $q$: a large prime that is used as the security parameter against the exhaustive search attack, for example, more than 160 bits [16];
- $G_1$: a multiplicative group of order $q$;
- $G_2$: an additive group of order $q$;
- $e$: $G_1 \times G_1 \rightarrow G_2$ is a bilinear map;
- $P$: a generator of $G_1$;
- $H_1, H_2: \{0,1\}^* \rightarrow G_i$ are cryptographic hash functions;
- $H_3: \{0,1\}^* \rightarrow Z_q^*$ is cryptographic hash function.

After that, $PKG$ randomly chooses an integer $s \in Z_q^*$ as his/her private key and computes the corresponding public key $Q = sP$. Finally, $PKG$ publishes $\{q, G_1, G_2, e, P, Q, H_1, H_2, H_3\}$ and keeps $\{s\}$ secret.

**Key Generation.** Let $ID_i$ be the identity and used as the public key for $U_i$ (for $i=0, 1, 2, \ldots, n$). In this phase, $PKG$ first generates the private key $S_i = sH_1(ID_i)$ for each $U_i$ and then deliver it to $U_i$ via a secure channel.

**Delegation.** The original signer $U_0$ first makes a warrant $w$ to transfer his/her signing power to the proxy signers $U_1, U_2, \ldots, U_n$. The warrant $w$ contains the identity of the original signer $ID_0$, the identities of the proxy signers $ID_1, ID_2, \ldots, ID_n$, and the start-time
$T_{\text{start}}$ and the end-time $T_{\text{end}}$ of the delegation. That is, $w = \{ID_0, ID_1, ID_2, \ldots, ID_n, T_{\text{start}}, T_{\text{end}}\}$. After that, $U_0$ generates a signature $\sigma_0 = (R_0, V_0)$ for $w$ by computing:

$$R_0 = r_0P$$

$$V_0 = h_0S_0 + r_0Q$$  \hspace{1cm} (1)

where $r_0 \in \mathbb{Z}_q^*$ is randomly chosen and $h_0 = H_3(ID_0, w, R_0)$. Finally, $U_0$ sends $\{ID_0, w, \sigma_0\}$ to each proxy signer $U_i$. Each proxy signer $U_i$ can verify the validity of the signature $\sigma_i = (R_i, V_i)$ of $w$ by checking the following equality:

$$e(V_i, P) = e(H_3(ID_i, w, R_i)H_3(ID_0) + R_0), Q)$$  \hspace{1cm} (3)

**Proxy Signature Generation.** When the proxy signer $U_i$ wants to sign the message $m_i$ at the time $T_i$ under the delegation specified in the warrant $w$, he/she first verifies whether $T_{\text{start}} \leq T_i \leq T_{\text{end}}$ or not. If $T_i$ is out of the valid period of the delegation, then abort this phase. Otherwise, $U_i$ generates an individual proxy signature $\sigma_i = (R_i, V_i)$ for $m_i$ by computing:

$$R_i = r_iP$$

$$V_i = V_0 + h_iS_i + r_iH_3(w)$$  \hspace{1cm} (4)

where $r_i \in \mathbb{Z}_q^*$ is randomly chosen and $h_i = H_3(ID_i, m_i, T_i, R_0)$. After that, $U_i$ sends $\{ID_i, m_i, T_i, \sigma_i\}$ to the aggregator $AGG$.

**Aggregation.** Upon receiving $\{ID_i, m_i, T_i, \sigma_i\}$ sent by $U_i$, the aggregator $AGG$ first verifies whether $T_{\text{start}} \leq T_i \leq T_{\text{end}}$ or not. If $T_i$ is out of the valid period of the delegation, then discard the proxy signature $\sigma_i$. Otherwise, $AGG$ ensures the validity of the individual proxy signature $\sigma_i = (R_i, V_i)$ of $m_i$ by checking the following equality:

$$e(V_i, P) = e(B_0 + h_iH_1(ID_i), Q)e(H_3(w), R_i)$$  \hspace{1cm} (6)

where $B_0 = H_3(ID_0, w, R_0)H_3(ID_0) + R_0$, $h_i = H_3(ID_i, m_i, T_i, R_0)$. For all accepted individual proxy signatures $\sigma_i$’s, $AGG$ generates an aggregate proxy signature $\sigma_{AM} = (AR, AV)$ for $AM = \{m_1, m_2, \ldots, m_n\}$ with respect to $AID = \{ID_1, ID_2, \ldots, ID_n\}$ and the individual signing time $AIT = \{T_1, T_2, \ldots, T_n\}$, where $AR = \sum_{i=1}^n R_i$ and $AV = \sum_{i=1}^n V_i$. Finally, $AGG$ publishes $\{ID_0, w, \sigma_0, AID, AIT, AM, \sigma_{AM}\}$ to the verifier(s).

**Aggregate proxy signature Verification.** Upon receiving $\{ID_0, w, \sigma_0, AID, AIT, AM, \sigma_{AM}\}$, the verifier $VER$ first verifies whether $T_{\text{start}} \leq T_i \leq T_{\text{end}}$ or not for all $T_i$’s. If any $T_i$ is out of the valid period of the delegation, then decline the aggregate proxy signature $\sigma_{AM}$. Otherwise, $VER$ ensures the validity of the aggregate proxy signature $\sigma_{AM} = (AR, AV)$ of $AM = \{m_1, m_2, \ldots, m_n\}$ by checking the following equality:

$$e(AV, P) = e(nB_0 + \sum_{i=1}^n (h_iH_1(ID_i))), Q)e(H_3(w), AR)$$  \hspace{1cm} (7)

where $B_0 = H_3(ID_0, w)H_3(ID_0) + R_0$, and $h_i = H_3(ID_i, m_i, T_i, R_0)$.

In the following, we will show that our proposed scheme works correctly. The correctness proofs include the verification of the delegation of the original signer, and the
verification of the individual proxy signatures and the aggregate proxy signature.

**Theorem 1.** The signature \( \sigma_0 = (R_0, V_0) \) for the warrant \( w \) issued from the original signer \( U_0 \) can be successfully verified by Eq. (3).

**Proof.** Recall that \( S_0 = \mathcal{S}(1) \), \( Q = sP \), and \( h_0 = H_2(ID_0, w, R_0) \). From Eqs. (1) and (2), we have

\[
e(V_0, P) = e(h_0S_0 + r_0Q, P) \quad \text{by Eq. (2)}
\]

\[
= e(H_2(ID_0, w)H_1(ID_0) + r_0sP, P) \quad \text{by Eq. (1)}
\]

\[
= e(H_2(ID_0, w)H_1(ID_0) + R_0, Q)
\]

which implies Eq. (3). Q.E.D.

**Theorem 2.** The individual proxy signature \( \sigma_i = (R_i, V_i) \) for the message \( m_i \) generated by the proxy signer \( U_i \) can be successfully verified by Eq. (6).

**Proof.** Recall that \( S_i = \mathcal{S}(1) \), \( B_0 = H_2(ID_0, w, R_0)H_1(ID_0) + R_0 \), and \( h_i = H_2(ID_i, m_i, T_i, R_0) \). From Eqs. (1), (2), (4), and (5), we have

\[
e(V_i, P) = e(V_0 + h_iS_i + r_0W, P) \quad \text{by Eq. (5)}
\]

\[
= e(h_0S_0 + r_0Q + h_iS_iH(ID_i), P) e(r_0W, P) \quad \text{by Eq. (2)}
\]

\[
= e(h_0sH_1(ID_i) + r_0sP + h_iS_iH(ID_i), P) e(W, r_0P) \quad \text{by Eq. (4)}
\]

\[
= e(s(H_2(ID_0, w, R_0)H_1(ID_0) + r_0sP + h_iH(I_i)), P) e(W, R_i) \quad \text{by Eq. (1)}
\]

\[
= e(H_2(ID_0, w, R_0)H_1(ID_0) + R_0 + h_iH_1(ID_i), sP) e(W, R_i)
\]

\[
e(B_0 + h_iH(ID_i), Q)e(W, R_i)
\]

which implies Eq. (6). Q.E.D.

**Theorem 3.** The aggregate proxy signature \( \sigma_A = (AR, AV) \) for the aggregate messages \( AM = \{m_1, m_2, ..., m_n\} \) generated by the proxy signers \( U_1, U_2, ..., U_n \) can be successfully verified by Eq. (7).

**Proof.** Recall that \( AR = \sum_{i=1}^{n} R_i \) and \( AV = \sum_{i=1}^{n} V_i \). From Eqs. (1), (2), (4), and (5), we have

\[
e(AV, P) = e(\sum_{i=1}^{n}(V_0 + h_iS_i + r_iH_1(w)), P) \quad \text{by Eq. (5)}
\]

\[
= e(\sum_{i=1}^{n}(h_0S_0 + r_0Q + h_iS_i), P)e(\sum_{i=1}^{n}r_iH_1(w), P) \quad \text{by Eq. (2)}
\]

\[
= e(\sum_{i=1}^{n}(h_0sH_1(ID_i) + r_0sP + h_iS_iH_1(ID_i)), P)e(H_1(w), \sum_{i=1}^{n}r_iP)
\]

\[
= e(s(H_2(ID_0, w, R_0)H_1(ID_0) + r_0sP + h_iH_1(ID_i)), P)e(H_1(w), \sum_{i=1}^{n}R_i) \quad \text{by Eq. (4)}
\]

\[
= e((\sum_{i=1}^{n}h_iH_1(ID_i) + R_0 + h_iH_1(ID_i)), sP)e(H_1(w), AR) \quad \text{by Eq. (1)}
\]

\[
= e((nB_0 + \sum_{i=1}^{n}(h_iH_1(ID_i))), Q)e(H_1(w), AR)
\]

which implies Eq. (7). Q.E.D.

### 3. SECURITY MODEL AND SECURITY ANALYSES

In this section, we first sketch a formal security model for the aggregate proxy sig-
nature scheme, and then show that our proposed scheme can withstand the chosen message attacks under the CDH assumption [12].

3.1 Security Model

Derived from the previous definitions regarded to the proxy signatures and the aggregate signatures addressed in [1-7, 9-11, 13-14], we sketch the security model for the aggregate proxy signature scheme as follows. Similarly to the formal models defined in [13, 14], we divide the potential adversary into the following three types:

**Type I adversary:** The adversary $A_I$ knows the public keys of the original signer and all the proxy signers, and attempts to forge the delegation for a chosen warrant or to forge the aggregate proxy signature for some chosen aggregate messages.

**Type II adversary:** The adversary $A_{II}$ knows not only the public keys of the original signer and all the proxy signers, but also all the private keys of the proxy signers, and attempts to forge the delegation by directly forging a valid signature for a chosen warrant.

**Type III adversary:** The adversary $A_{III}$ knows not only the public keys of the original signer and all the proxy signers, but also the private key of the original signer, and attempts to forge the aggregate proxy signature for some chosen aggregate messages.

It is to see that if the aggregate proxy signature scheme can withstand the attacks plotted from both the Type II and the Type III adversaries, then it will be secure against the Type I adversary straightforwardly. As pointed out by Wu et al. [13] and Xu et al. [14], the delegation in a warrant based proxy signature scheme can be directly realized by the original signer’s signature on a warrant which contains the information regarding the proxy signers and their delegated signing powers to prevent the misuse of the warrant. Thereafter, we only focus on the discussion of the unforgeability of the warrant, the proxy signatures, and the aggregate signature. The security model on our proposed scheme is defined in the following.

**Definition 1.** We say an aggregate proxy signature scheme is secure against any Type II adversary if there is no probabilistic polynomial-time adversary $A_{II}$ can forge a valid signature $\sigma_w$ on a chosen warrant $w$ by playing the game with a challenger $C$, as depicted in Figure 2.

**Definition 2.** We say a Type II adversary $A_{II}$ - $(t, q_{H_1}, q_K, q_{H_2}, q_D, \varepsilon)$ can break our proposed aggregate proxy signature scheme if $A_{II}$ can run in time at most $t$, makes at most $q_{H_1}$ queries to $H_1$ query, at most $q_K$ queries to KeyGen query, at most $q_{H_2}$ queries to $H_2$ query, and at most $q_D$ queries to Delegation query, to win the game (depicted in Figure 2) with success probability at least $\varepsilon$.

**Definition 3.** We say an aggregate proxy signature scheme is secure against any Type III adversary if there is no probabilistic polynomial-time adversary $A_{III}$ can forge a valid aggregate proxy signature $\sigma_{AM}$ on the chosen aggregate messages $AM = \{m_1, m_2, \ldots, m_n\}$.
by playing the game with a challenger $C$, as depicted in Figure 3.

**Definition 4.** We say a Type III adversary $A_{III}$-$(t, q_{H_1}, q_{K}, q_{H_2}, q_{H_3}, q_{S}, n, \epsilon)$ can break our proposed aggregate proxy signature scheme if $A_{III}$ runs in time at most $t$, makes at most $q_{H_1}$ queries to $H_1$ query, at most $q_{K}$ queries to KeyGen query, at most $q_{H_2}$ queries to $H_2$ query, at most $q_{H_3}$ queries to $H_3$ query, and at most $q_{S}$ queries to AggSign query for obtaining at most $n$ forged individual proxy signatures, to win the game (depicted in Figure 3) with success probability at least $\epsilon$.

**Setup.** $C$ runs System Setup phase to obtain system parameters.

**H_1 query.** $C$ runs $H_1$ function on a chosen identity $ID_i$ and returns $H_1(ID_i)$.

**KeyGen query.** $C$ runs Key Generation phase on the given identity $ID_i$ and returns a private key $S_i$ associated to $ID_i$.

**H_2 query.** $C$ runs $H_2$ function on a chosen identity $ID_i$, a chosen warrant $w$, and a random $R \in G_1$, and then returns $H_2(ID_i, w, R)$.

**Delegation query.** $C$ runs Delegation phase on a chosen warrant $w$ and returns a signature $\sigma_w$ of $w$.

**Output.** $A_{III}$ outputs $\{ID_0, w', \sigma'_w\}$ and wins the game if:

1. $w'$ is not $w$; and
2. $\sigma'_w$ is a valid signature of $w'$.

Figure 2. The game for Definition 1.

**Setup.** $C$ runs System Setup phase to obtain system parameters.

**H_1 query.** $C$ runs $H_1$ function on a chosen identity $ID_i$ and returns $H_1(ID_i)$.

**KeyGen query.** $C$ runs Key Generation phase on a given identity $ID_i$ and returns a private key $S_i$ associated to $ID_i$.

**H_2 query.** $C$ runs $H_2$ function on a chosen identity $ID_i$, a chosen warrant $w$, and a random $R \in G_1$, and then returns $H_2(ID_i, w, R)$.

**Delegation query.** $C$ runs Delegation phase on a given warrant $w$ and returns a signature $\sigma_w$ of $w$.

**Output.** $A_{II}$ outputs $\{ID_0, w', \sigma'_w\}$ and wins the game if:

1. $w'$ is not $w$; and
2. $\sigma'_w$ is a valid signature of $w'$.

Figure 3. The game for Definition 3.

3.2 Security Analyses
Before giving the security proofs, we first introduce the computational Diffie-Hellman problem in $G_1$, i.e., the CDH $G_1$ Problem [15], and its security assumption as below.

**CDH $G_1$ Problem:** Given $P, xP, yP, zP \in G_1$, for unknown $x, y \in \mathbb{Z}_q^*$, it is computationally infeasible to compute $xyzP$ for any probabilistic polynomial-time algorithm.

**CDH $G_1$-$(t', \varepsilon')$ Assumption:** The CDH $G_1$-$(t', \varepsilon')$ Assumption holds if there does not exist $t'$-time algorithm that has advantage $\varepsilon'$ in solving the CDH $G_1$ Problem.

In the following, we will show that our proposed scheme is secure against the chosen message attack under the CDH $G_1$-$(t', \varepsilon')$ Assumption.

**Theorem 4.** Let $t_{G_1}$ be the time of computing a scalar multiplication/inversion on $G_1$. If there exists an adversary $A_{II}$ - $(t, q_{H_1}, q_{K}, q_{H_2}, q_{D}, \varepsilon)$ who can break our proposed aggregate proxy signature scheme, then there exists a challenger $C$ who will face the CDH $G_1$-$(t', \varepsilon')$ Assumption with success probability $\varepsilon$ in time $t'$, where

$$\varepsilon \geq \frac{\varepsilon'(q_k + 1)}{(1 - \frac{1}{q_k + 1})^{t_{G_1} + 1}}$$

$$t \leq t' - t_{G_1} (q_{H_1} + q_K + 3q_{D} + 4)$$

**Proof.** We first show how to construct a challenger $C$ that can solve the CDH $G_1$ problem with success probability at least $\varepsilon'$ in time $t'$. Given $P, X = xP, Y = yP \in G_1$, for unknown $x, y \in \mathbb{Z}_q^*$, the challenger $C$ plays the following game with the adversary $A_{II}$ to output $xyP \in G_1$:

**Setup.** $C$ sets $PKG$’s public key $Q$ to be $X$, i.e., $X = Q (= sP)$, and starts the system parameters given by $A_{II}$. From then on, $A_{II}$ can make $H_1$ query, KeyGen query, $H_1$ query, and Delegation query with $C$ at any time.

**$H_1$ query.** For responding to this query, $C$ maintains a $H_1$-list of 4-tuples $<ID, L, a, b>$, where $L \in G_1, a \in \mathbb{Z}_q^*$, and $b \in \{0, 1\}$. Note that $H_1$-list is initially empty. Upon receiving $A_{II}$’s a query to the oracle $H_1$ on a given identity $ID_\mu$, the following steps are performed:

1. If $ID_\mu$ has already appeared on the $H_1$-list, then $C$ returns $L_\mu$ where $L_\mu = b_\mu a_\mu yP + (1 - b_\mu) a_\mu Y$.
2. Otherwise, $C$ generates a random coin $b_\mu \in \{0, 1\}$ so that Pr$[b_\mu = 0] = 1/(q_k + 1)$ and randomly chooses $a_\mu \in \mathbb{Z}_q^*$. If $b_\mu = 0$ holds, then $C$ computes $L_\mu = a_\mu Y$, else $L_\mu = a_\mu P$. Finally, $C$ adds the instance of 4-tuple $<ID_\mu, L_\mu, a_\mu, b_\mu>$ into the $H_1$-list and returns $L_\mu$.

**KeyGen query.** When $A_{II}$ requests a private key associated to a given identity $ID_\mu$, $C$ first runs $H_1$ query and then gets the corresponding instance of 4-tuple $<ID_\mu, L_\mu, a_\mu, b_\mu>$ from the $H_1$-list. If $b_\mu = 0$ holds, then $C$ returns failure and aborts, otherwise returns $a_\mu Q$. Note
that in this query, \( L_\mu = a_\mu P = H_1(ID_\mu) \) if \( h_\mu = 1 \) holds. This implies that the private key associated to \( ID_\mu \) is \( a_\mu Q = a_\mu P = sH_1(ID_\mu) \).

**H₂ query.** For responding to this query, \( C \) maintains a **H₂-list** of 4-tuples \(<ID, w, R, h>\), where \( w \) is a given warrant, \( R \in G_1 \), and \( h \in \mathbb{Z}_q^* \). Note that **H₂-list** is initially empty. Upon receiving \( A_I \)'s a query to the oracle \( H_2 \) on given \( ID_\mu, w_\mu \) and \( R_\mu \) the following steps are performed:

1. If \( ID_\mu, w_\mu \) and \( R_\mu \) have already appeared on the **H₂-list**, then \( C \) returns \( h_\mu \).
2. Otherwise, \( C \) randomly chooses an integer \( h_\mu \in \mathbb{Z}_q^* \) and adds the instance of 4-tuple \(<ID_\mu, w_\mu, R_\mu, h_\mu>\) into the **H₂-list** and returns \( h_\mu \).

**Delegation query.** Upon receiving \( A_I \)'s query on a given warrant \( w_\mu \), for an original signer with the identity \( ID_\mu \) \( C \) first confirms that \(<ID_\mu, w_\mu>\) was not requested before. If \(<ID_\mu, w_\mu>\) was requested before, then \( C \) returns failure and aborts, otherwise does the following:

1. Run **H₁ query** on \( ID_\mu \) and get the corresponding instance of 4-tuple \(<ID_\mu, L_\mu, a_\mu, h_\mu>\) from the **H₁-list**.
2. Compute \( R_\mu = r_\mu P \), where \( r_\mu \in \mathbb{Z}_q^* \) is randomly chosen.
3. Run **H₂ query** on \( ID_\mu, w_\mu \), and \( R_\mu \) and get the corresponding instance of 4-tuple \(<ID_\mu, w_\mu, R_\mu, h_\mu>\) from the **H₂-list**.
4. If \( h_\mu = 0 \) holds, then return failure and abort, else compute \( V_\mu = h_\mu \mu Q + r_\mu Q \) and return \( \sigma_\mu = (R_\mu, V_\mu) \) as a signature for \( w_\mu \). Note that \( e(h_\mu L_\mu + R_\mu Q, Q) = e(V_\mu, P) \).

**Output.** Finally, \( A_I \) halts and \( C \) returns either failure or a valid forged signature \( \sigma_\mu = (R_\mu, V_\mu) \) for the given warrant \( w_\mu \) associated to the identity \( ID_\mu \). If \( \sigma_\mu \) satisfies Eq. (3), then \( C \) constructs another signature \( \sigma'_\mu = (R_\mu, V'_\mu) \) for another chosen warrant \( w'_\mu \) in polynomial time, where \( V'_\mu = h'_\mu S_i + r_\mu Q \). To do this, \( C \) gets the corresponding instance of 4-tuple \(<ID_\mu, L_\mu, a_\mu, h_\mu>\) from the **H₁-list**. If \( h_\mu = 1 \) holds, then \( C \) returns failure and aborts, else computes \( xyP = (a_\mu(h_\mu - h'_\mu))^{-1}(V_\mu - V'_\mu) \) and outputs \( xyQ \) as a solution to the CDH \( G_1 \) problem. Note that \( V_\mu - V'_\mu = (h_\mu - h'_\mu)S_i = (h_\mu - h'_\mu) sH_1(ID_\mu) = (h_\mu - h'_\mu)x(a_\mu P), \) where \( sP = Q = X = xP \) and \( H_1(ID_\mu) = L_\mu = a_\mu Y = a_\mu P \).

Next, we show that \( C \) can solve the CDH \( G_1 \) problem with the success probability at least \( \varepsilon' \). There are three events considered:

- **E₁**: \( C \) does not abort any \( A_I \)'s request of **Delegation query**.
- **E₂**: \( A_I \) successfully forges a signature for a chosen warrant \( w_\mu \) with the identity \( ID_\mu \).
- **E₃**: \( C \) does not abort **Output** and event \( E₂ \) occurs at the same time.

It is to see that \( C \) can successfully forge the signature for any chosen warrant if all the above events happen. The probability of \( Pr[E₁ \land E₂ \land E₃] \) can be further decomposed as

\[
Pr[E₁ \land E₂ \land E₃] = Pr[E₁] Pr[E₂|E₁] Pr[E₃|E₁ \land E₂]
\]

**Claim 4.1.** \( C \) does not abort any \( A_I \)'s request of **Delegation query** with the probability
at least \((1-1/(q_k + 1))^8\)). Hence, the probability of event \(E_1\) is \(Pr[E_1] \geq (1-1/(q_k + 1))^8\).

**Claim 4.2.** If \(C\) does not abort any \(A_i\)'s request of Delegation query, then \(A_i\)'s view is identical to its view in the real attack. Hence, the probability of event \(E_2\) is \(Pr[E_2|E_1] \geq \epsilon\).

**Claim 4.3.** \(C\) does not abort and \(A_0\) outputs a valid signature of a chosen warrant with the probability at least \((1-1/(q_k + 1))/(q_k + 1)\). Hence, the probability of event \(E_3\) is \(Pr[E_3|E_1 \wedge E_2] \geq (1-1/(q_k + 1))/(q_k + 1)\).

To complete the proof of Theorem 4, the probability that \(C\) produces the correct outputs is at least \(\epsilon(1-1/(q_k + 1))^{8\alpha + 1}/(q_k + 1) \geq \epsilon\) under Eq. (8). Detailed proofs for Claims 4.1, 4.2, and 4.3 are given in Appendix. One can see that each \(H_1\) query and KeyGen query requires one scalar multiplication in \(G_1\), and each Delegation query requires 3 scalar multiplications in \(G_1\). As to Output for each game, \(C\) requires 3 scalar multiplications and one inversion in \(G_1\) to obtain the solution of the CDH\(_{G_1}\) problem. Hence, the total running time is at most \(t + t_G(q_h + q_k + 3q_D + 4) \leq t'\). Q.E.D.

**Theorem 5.** Let \(t_G\) be the time of computing a scalar multiplication or inversion on \(G_1\). If there exists an adversary \(A_{III} (t, q_h, q_k, q_{h_3}, q_{h_3}, q_s, n, \epsilon)\) who can break our proposed aggregate proxy signature scheme then there exists a challenger \(C\) who will face the CDH\(_{G_1}\) and \(\epsilon'\).'s same as those described in the proof of Theorem 4.

**Setup, \(H_1\) query, and KeyGen query** are the same as those described in the proof of Theorem 4.

**\(H_2\) query.** For responding to this query, \(C\) maintains a \(H_2\)-list of 5-tuples \(<ID, m, T, R, h>\), where \(m\) is a given message, \(T\) is a given valid signing time, \(R \in G_1\), and \(h \in Z_q\). Note that \(H_2\)-list is initially empty. Upon receiving \(A_{III}'s\) a query to the oracle \(H_2\) on given \(ID_{m'}\) \(m_{m'}\) \(T_{m'}\) and \(R_{m'}\) the following steps are performed:

1. If \(ID_{m'}\) \(m_{m'}\) \(T_{m'}\) and \(R_{m'}\) have already appeared on the \(H_2\)-list, then \(C\) returns \(h_{m'}\).
2. Otherwise, \(C\) randomly chooses an integer \(h_{m'} \in Z_q\) and adds the instance of 5-tuple \(<ID_{m'}, w_{m'}, T_{m'}, R_{m'}, h_{m'}>\) into the \(H_2\)-list and returns \(h_{m'}\).

**\(H_3\) query.** For responding to this query, \(C\) maintains a \(H_3\)-list of 4-tuples \(<w, \Omega, \alpha, \beta>\),
where $w$ is a chosen warrant, $\Omega \in G_1$, $\alpha \in Z_q^*$, and $\beta \in \{0, 1\}$. Note that the $H_f$-list is initially empty. Upon receiving $A_{III}$’s query to the oracle $H_3$ on a chosen warrant $w_\mu$, the following steps are performed:

1. If $w_\mu$ has already appeared on the $H_f$-list, then $C$ returns $\Omega_\mu$. Note that $\Omega_\mu = \beta_0\alpha_0Q + (1 - \beta_0)\alpha_0Y$.

2. Otherwise, $C$ randomly chooses $\alpha_\mu \in Z_q^*$ and generates a random coin $\beta_\mu \in \{0, 1\}$ so that $\Pr[\beta_\mu = 0] = 1/(q+1)$. If $\beta_\mu = 0$ holds, then $C$ computes $\Omega_\mu = \alpha_\muY$; else $\Omega_\mu = \alpha_\muQ$. Finally, $C$ adds the instance of 4-tuple $<w_\mu, \Omega_\mu, \alpha_\mu, \beta_\mu>$ into the $H_f$-list and returns $\Omega_\mu$.

**AggSign query.** By definition, $A_{III}$ knows the private key of the original signer $U_0$ and has the ability to generate a forged valid signature $\sigma_0 = (R_0, V_0)$ for a chosen warrant $w^*$. When $A_{III}$ makes this query on aggregate messages $AM = \{m_1, m_2, \ldots, m_n\}$ for $n$ proxy signers with the identities $AID = \{ID_1, ID_2, \ldots, ID_n\}$ and the signing time $AIT = \{T_1, T_2, \ldots, T_n\}$ under the chosen warrant $w_\mu$, $C$ first confirms that $\{AID, AIT, AM\}$ has not been requested before. If $\{AID, AIT, AM\}$ was requested before, then $C$ returns failure and aborts, otherwise does the following on each $ID_\mu$ and $m_\mu$ (for $\mu = 1, 2, \ldots, n$):

1. Run $H_i$ query on $ID_\mu$ and get the corresponding instance of 4-tuple $<ID_\mu, L_\mu, a_\mu, b_\mu>$ from the $H_f$-list.

2. Run $H_q$ query on $ID_\mu, m_\mu, T_\mu$, and $R_\mu$, and get $h_\mu$ from the $H_q$-list.

3. Run $H_q$ query on $w^*$, and get the corresponding instance of 4-tuple $<w^*, \Omega_\mu, \alpha_\mu, \beta_\mu>$ from the $H_f$-list.

4. If $b_\mu = \beta_\mu = 0$ holds, then return failure and abort.

5. If $b_\mu = 0$ and $\beta_\mu = 1$ holds, then compute $R_\mu = r_\mu P - \alpha_0^{-1}h_\mu L_\mu$ and $V_\mu = V_0 + r_\mu Q$, where $r_\mu \in Z_q^*$ is randomly chosen and $V_0 = H_3(ID_\mu, w^*, R_\mu)S_\mu + sR_\mu$. Note that $e(H_2(ID_\mu, w^*, R_\mu)H_3(ID_\mu) + R_\mu + h_\mu L_\mu, Q)e(\Omega_\mu, R_\mu) = (V_\mu, P)$.

6. If $b_\mu = 1$ holds, then compute $R_\mu = r_\mu P$ and $V_\mu = V_0 + h_\mu L_\mu + r_\mu Q$, where $r_\mu \in Z_q^*$ is randomly chosen. Note that $e(H_2(ID_\mu, w^*, R_\mu)H_3(ID_\mu) + R_\mu + h_\mu L_\mu, Q)e(\Omega_\mu, R_\mu) = e(V_\mu, P)$.

If $C$ does not abort any one of the queries and successfully outputs $n$ forged individual proxy signatures $(R_\mu, V_\mu)$ for $m_\mu$, then $C$ computes $AR = \sum_{\mu=1}^n R_\mu$ and $AV = \sum_{\mu=1}^n V_\mu$, and returns $(AR, AV)$.

**Output.** Finally, $A_{III}$ halts and $C$ returns either failure or a valid forged aggregate proxy signature $\sigma'_{AM} = (AR, AV)$ for the given aggregate messages $AM = \{m_1, m_2, \ldots, m_n\}$ associated to the proxy signers with the identities $AID = \{ID_1, ID_2, \ldots, ID_n\}$ and the signing time $AIT = \{T_1, T_2, \ldots, T_n\}$. If $\sigma_{AM}$ satisfies Eq. (7), then $C$ constructs another aggregate proxy signature $\sigma'' = (AR, AV'')$ for another chosen aggregate messages $AM' = \{m'_1, m'_2, \ldots, m'_n\}$ in polynomial time, where $m'_\mu$ is not any of $m_\mu$ and $AV'' = \sum_{\mu=1}^n (V_0 + h'_\mu S_\mu + r'_W)$. To do this, $C$ gets the corresponding instances of 4-tuples $<ID_\mu, L_\mu, a_\mu, b_\mu>$ from the $H_f$-list for each $ID_\mu$ (for $\mu = 1, 2, \ldots, n$), and gets the corre-
YEN-CHING LIN, TZONG-CHEN WU, AND JIA-LUN TSAI

sponding instances of 4-tuple \(<w^{*}, \Omega_0, \alpha_0, \beta_0>\) from the \(H_3\)-list. If any \(b_\mu = 1\) or \(\beta_0 = 1\), then \(C\) returns failure and aborts. Otherwise, \(C\) computes \(xyP = (\sum_{\mu=1}^{n}a_\mu(h_\mu - h'_\mu)^{-1}(AV - AV'))\) and outputs \(xyP\) as a solution to the CDH$_{G_1}$ problem.

Note that \(AV - AV' = \sum_{\mu=1}^{n}(h_\mu - h'_\mu)S_\mu = \sum_{\mu=1}^{n}(h_\mu - h'_\mu)x(a_\mu yP)\).

Next, we show that \(C\) can solve the CDH$_{G_1}$ problem with the success probability at least \(\epsilon'\). There are three events considered:

- **\(E_1\):** \(C\) does not abort for any \(A_{III}\) 's request of \(\text{AggSign query}\).
- **\(E_2\):** \(A_{III}\) successfully forges a aggregate proxy signature for the chosen aggregate messages \(AM = \{m_1, m_2, \ldots, m_n\}\) with the proxy signers' identities \(AID = \{ID_1, ID_2, \ldots, ID_n\}\) and the signing time \(AIT = \{T_1, T_2, \ldots, T_n\}\).
- **\(E_3\):** \(C\) does not abort \(Output\) and event \(E_2\) occurs at the same time.

It is to see that \(C\) can successfully forge the aggregate proxy signature for any chosen aggregate message if all the above events happen. The probability of \(\Pr[E_1 \wedge E_2 \wedge E_3]\) can be further decomposed as Eq. (8).

**Claim 5.1.** \(C\) does not abort any request of \(A_{III}\) 's \(\text{AggSign query}\) with the probability at least \((1 - (1/(qK + 1)(qS + 1)))^{\text{qH_3}}\). Hence, the probability of event \(E_1\) is \(\Pr[E_1] \geq (1 - (1/(qK + 1)(qS + 1)))^{\text{qH_3}}\).

**Claim 5.2.** If \(C\) does not abort any request of \(A_{III}\) 's \(\text{AggSign query}\), then \(A_{III}\)'s view is identical to its view in the real attack. Hence, the probability of event \(E_2\) is \(\Pr[E_2|E_1] \geq \epsilon\).

**Claim 5.3.** \(C\) does not abort after \(A_{III}\) outputs a valid aggregate proxy signature with the probability at least \((1 - 1/(qK + 1)(qS + 1)))^{\text{qH_3}}\). Hence, the probability of event \(E_3\) is \(\Pr[E_3|E_1 \wedge E_2] \geq (1 - 1/(qK + 1)(qS + 1)))^{\text{qH_3}}\).

To complete the proof of Theorem 5, the probability that \(C\) produces the correct outputs is at least \(\epsilon(1 - 1/(qK + 1)(qS + 1)))^{\text{qH_3}}\) under Eq. (8). Detailed proofs for Claims 5.1, 5.2, and 5.3 are given in Appendix. One can see that each \(H_1\) query, \(\text{KeyGen query}\), and \(H_2\) query requires one scalar multiplication in \(G_1\), and each \(\text{AggSign query}\) requires at most 3n scalar multiplications in \(G_1\). As to \(Output\) for each game, \(C\) requires 2n + 1 scalar multiplications and one inversion in \(G_1\) to obtain the solution of the CDH$_{G_1}$ problem. Hence, the total running time is at most \(t + t_{G_1}(q_{H_1} + q_{K} + q_{H_3} + 3nq_{S} + 2n + 2) \leq t'\). Q.E.D.

### 4. PERFORMANCE ANALYSES

We measure the performance of our proposed scheme with respect to the required computational complexity and the communication cost. Computational complexity is measured by the required bilinear pairing operations, scalar multiplications and additions of points in \(G_1\), and the computation of hash functions. Communication cost is measured by the size of transmitted messages in each phase. Interested reader may refer [17, 18]...
for more detailed implementation results about bilinear pairing operations. It can be seen that bilinear pairing operation is more time-consuming than that of point operation and some existing hash function in practice. We use the following symbols for evaluating the performance of our proposed scheme:

- \( TGE \): the time required to perform one bilinear pairing operation \( e \);
- \( TGM \): the time required to perform one scalar multiplication of point in \( G_1 \);
- \( TGA \): the time required to perform one point addition in \( G_1 \);
- \( TH_1 \): the time required to compute the \( H_1 \) function;
- \( TH_2 \): the time required to compute the \( H_2 \) function;
- \( TH_3 \): the time required to compute the \( H_3 \) function; and
- \( |x| \): the size of integer or string \( x \).

Note that in the proposed scheme, the time required to verify the validity of signing time specified in the warrant is negligible to the required time to perform pairing operations or point calculation in \( G_1 \). We ignore it in the computational complexity of performance evaluation. Table 1 shows the analyses of computational complexity and communication cost of our proposed scheme. Note that our proposed scheme requires constant bilinear pairing operations for signature verification, regardless the number of participant proxy signers has involved. Besides, the size of aggregate proxy signature is the same as each of individual proxy signature.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Computational Complexity</th>
<th>Communication cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Setup</td>
<td>( TGM + TH_1 + TGM )</td>
<td>( 3</td>
</tr>
<tr>
<td>Key Generation</td>
<td></td>
<td>(</td>
</tr>
<tr>
<td>Delegation</td>
<td>For warrant signature generation: ( 3TGM + TGA + TH_2 )</td>
<td>(</td>
</tr>
<tr>
<td></td>
<td>For warrant signature verification: ( 2TGE + TGM + TH_1 + TH_2 )</td>
<td></td>
</tr>
<tr>
<td>Proxy Signature Generation</td>
<td>For each proxy signature generation: ( 3TGM + 2TGA + TH_1 + TH_2 + TH_3 )</td>
<td>(</td>
</tr>
<tr>
<td>Aggregation</td>
<td>For each proxy signature verification: ( 3TGE + 3TGM + 3TGA + TH_2 + TH_1 + TH_3 )</td>
<td>( n(</td>
</tr>
<tr>
<td>Aggregate Proxy Signature Verification</td>
<td>For aggregating proxy signatures: ( 2(n-1)TGA )</td>
<td></td>
</tr>
</tbody>
</table>

Note: \( n \) is the number of participant proxy signers.

5. CONCLUSIONS

We have presented an ID-based aggregate proxy signature scheme that realizes a warrant-based delegation, where an original signer can delegate his/her signing authority to a set of \( n \) proxy signers and allow distinct proxy signers to sign distinct messages under the warrant, respectively, in such a way that these \( n \) individual proxy signatures can be aggregated into a single one without expansion. Furthermore, the verifier only
needs to know the public key of the Private Key Generator and the identities of the original signer and the proxy signers for verifying the aggregate proxy signature. Our proposed scheme requires constant bilinear pairing operations for signature verification. Besides, the size of aggregate proxy signature is the same as that of each of the individual proxy signatures, regardless of the number of participant proxy signers has involved. We have shown that our proposed scheme is secure against the chosen message attacks under the CDH assumption.

REFERENCES

APPENDIX

Proof for Claim 4.1. It is reasonable to assume that \( A_{II} \) does not request Delegation query for the same warrant twice. \( C \) gets the instance of the corresponding 4-tuple \(<ID_{\mu}, L_{\mu}, a_{\mu}, b_{\mu}=0>\) from the \( H_1 \)-list with the probability \( 1/(q_K + 1) \). It is to see that the probability that \( C \) does not abort Delegation query and returns a valid signature is \( 1-(1/\left(q_K + 1\right)) \). If \( A_{II} \) makes \( q_D \) queries to Delegation query, then the probability that \( C \) does not abort for all these \( q_D \) queries is \( \left(1-(1/(q_K + 1))\right)^{q_D} \). That is, the probability of event \( E_1 \) is at most \( \left(1-(1/(q_K + 1))\right)^{q_D} \).

Proof for Claim 4.2. \( C \) will output a valid delegation signature \((R, V)\) on the chosen warrant \( w \) in the Delegation query only if he/she knows the instance value of \( a_{\mu} \) satisfying \( a_{\mu}P = sH(ID_{\mu}) \) from the \( H_1 \)-list. However, the results of KeyGen query are viewed as in the real attack, which implies that each output of KeyGen query is uniformly distributed in \( G_1 \). That is, the probability of \( \Pr[E_2|E_1] \) is at least \( \varepsilon \).

Proof for Claim 4.3. Under the conditions that both \( E_1 \) and \( E_2 \) have occurred, \( C \) does not abort Output after \( A_{II} \) has generated a valid signature on a chosen warrant \( w \). \( C \) gets the instance of the corresponding 4-tuple \(<ID_{\mu}, L_{\mu}, a_{\mu}, b_{\mu}=0>\) from the \( H_1 \)-list with the probability \( 1/(q_K + 1) \). Thus, the probability that \( C \) does not abort Output is \( 1/(q_K + 1) \). Recall Claim 4.1, the probability that \( C \) returns a valid signature of \( w \) is \( (1-(1/(q_K + 1))) \). This implies that the probability of \( \Pr[E_3|E_1 \land E_2] \) is at least \( \left(1-(1/(q_K+1))\right)/(q_K+1) \).

Proof for Claim 5.1. It is reasonable to assume that \( A_{III} \) does not request AggSign query for the same aggregate messages twice. \( C \) gets the instance of the corresponding 4-tuple
<ID, L, a, b =0> from the $H_1$-list with probability $1/(q_k + 1)$ and the instance of the corresponding 4-tuple $<w, \Omega, a, \beta =0>$ from the $H_1$-list with probability $1/(q_s + 1)$. Therefore, the probability that $C$ does not abort $\text{AggSign}$ query and generates a valid individual proxy signature is $1-1/(q_k + 1)(q_s + 1)$. It is to see that the probability that $C$ returns a forged aggregate proxy signature with $n$ proxy signers’ identities is $(1-1/(q_k + 1)(q_s + 1))^n$. If $A_{III}$ makes $q_s$ queries to $\text{AggSign}$ query, then the probability that $C$ does not abort for all these $q_s$ queries is $(1-1/(q_k + 1)(q_s + 1))^n$. That is, the probability of event $E_1$ is at most $(1-1/(q_k + 1)(q_s + 1))^n$.

Proof for Claim 5.2. $C$ will output a valid aggregate proxy signature $(AR, AV)$ on the chosen aggregate messages $AM = \{m_1, m_2, ..., m_n\}$ in the $\text{AggSign}$ query only if he/she knows the instance value of $a$ satisfying $aP = H(ID)$ from the $H_1$-list or $a$ satisfying $aQ = H(w)$ from the $H_3$-list. However, the results of $\text{KeyGen}$ query and $H_3$ query are viewed as in the real attack, which implies that each output of $\text{KeyGen}$ query or $H_3$ query is uniformly distributed in $G_1$. That is, the probability of $\Pr[E_2|E_1]$ is at least $\epsilon$.

Proof for Claim 5.3. Under the conditions that both $E_1$ and $E_2$ have occurred, $C$ does not abort $\text{Output}$ after $A_{III}$ has generated a valid aggregate proxy signature on the chosen aggregate messages $AM = \{m_1, m_2, ..., m_n\}$. $C$ gets the instance of the corresponding 4-tuple $<ID, L, a, b =0>$ from the $H_1$-list with the probability $1/(q_k + 1)$ and gets the instance of the corresponding 4-tuple $<w, \Omega, a, \beta =0>$ from the $H_1$-list with the probability $1/(q_s + 1)$. Therefore, the probability that $C$ does not abort $\text{Output}$ is $1/(q_k + 1)(q_s + 1)$. Recall Claim 5.1, the probability that $C$ returns a valid aggregate proxy signature of $AM$ is $(1-1/(q_k + 1)(q_s + n))^n$. This implies that the probability of $\Pr[E_3|E_1 \land E_2]$ is at least $(1-1/(q_k + 1)(q_s + 1))^n/(q_k + 1)(q_s + 1)$.
Tzong-Chen Wu (吳宗成) received the B.S. degree in Information Engineering from National Taiwan University in 1983, M.S. degree in Applied Mathematics from National Chung Hsing University in 1989, and Ph.D. degree in Computer Science and Information Engineering from National Chiao Tung University in 1992. Prof. Wu joined the Department of Information Management, National Taiwan University of Science and Technology (NTUST) in 1992. Since February 1997 till now, he has been the professor in the Department of Information Management, NTUST. Prof. Wu is the members of IEEE, ACM, and the Chinese Cryptology and Information Security Association (CCISA). His current research interests include data security, cryptography, network security, and data engineering.

Jia-Lun Tsai (蔡佳倫) is a Ph.D. student in the Department of Information Management, National Taiwan University of Science and Technology (NTUST). His research interests include cryptography, wireless security, and network security.