A new identity based proxy signature scheme

Bin Wang

Information Engineering College of Yangzhou University

Yangzhou City, Jiangsu Province, P.R.China

Tel: 086-0514-82220820

E-mail:xiaobinw@yahoo.com

Abstract: Proxy signature schemes allow a proxy signer to generate proxy signatures on behalf of an original signer. Mambo et al. first introduced the notion of proxy signature and a lot of research work can be found on this topic nowadays. Recently, many identity based proxy signature schemes were proposed. However, some schemes are vulnerable to proxy key exposure attack. In this paper, we propose a security model for identity based proxy signature schemes. Then an efficient scheme from pairings is presented. The presented scheme is provably secure in the random oracle model. In particular, the new scheme is secure against proxy key exposure attack.

Keywords: Digital signature, Proxy signature, Identity based signature, Bilinear pairing, Random oracle model;

1. Introduction

Digital signature is one of the most important security services under the electronic commercial environment. As a variation of ordinary digital signature schemes, the notion of proxy signature was first introduced by Mambo, Usuda and Okamoto [10] in 1996. In a proxy signature scheme, an original signer is allowed to delegate his signing capability to a designated proxy signer. Then the proxy signer can generate proxy signatures on behalf of the original signer. Proxy signatures have found numerous practical applications. Examples include distributed systems, Grid computing, and mobile communication.

There are three types of delegation: full delegation, partial delegation and delegation by warrant. In a full delegation scheme, the original signer gives his secret signing key to the proxy signer as the proxy signing key. Hence, for a given message, signatures generated between the original signer and the proxy signer are indistinguishable. In a partial delegation

scheme, the proxy signing key is derived from the original signer's secret key. In addition, it is computationally infeasible for the proxy signer to derive the original signer's secret key. However, the messages that a proxy signer can sign are not limited. In a delegation by warrant scheme, the original signer creates and signs a warrant that is used to certify the legitimacy of the proxy signer.

Since Mambo et al. introduced the concept of proxy signature, several kinds of proxy signature schemes have been proposed [8,9]. Furthermore, there are various extensions of the proxy signature primitive, such as threshold proxy signature [6], proxy multi-signature signature [7], etc. Informally, the basic security properties for proxy signature schemes can be described as follows [9]:

Verifiability: From a proxy signature, a verifier can be convinced of the original signer's agreement on the signed message.

Unforgeability: Only the designated proxy signer can generate a valid proxy signature on behalf of the original signer.

Strong identifiability: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.

Undeniability: The designated proxy signer cannot deny a valid proxy signature generated by him.

Prevention of misuse: A proxy signing key cannot be used for purpose other than generating valid proxy signatures.

The definition of secure digital signature schemes was given by Goldwasser, Micali and Rivest in [4]. However, the first work [1] on proxy signature in the provable security direction was done by Boldyreva, Palacio and Warinschi in 2003. They formalized the notion of security for proxy signature schemes in order to prove the security of proxy signature schemes under some well-established hard problems. However, Jacob C.N. Schdult et al. [12] pointed out that the scheme proposed in [1] is vulnerable to proxy key exposure attack. That is, upon exposure of a proxy signing key, an adversary can recover the private key of the corresponding proxy signer.

In the conventional Public Key Infrastructure (**PKI**), the binding between a user's public key and the user's identity is obtained via certificates issued by some trusted **CA**s. However,

PKI is considered to be costly to deploy and maintain. In [11], Shamir introduced the notion of identity based public key cryptography (ID-PKC) in which the public key would be the user's unique identity. The motivation of ID-PKC is to simplify certificate management. The advantage of ID-based schemes is that no certificate information and public key verification is needed. Since then, many identity based cryptosystems are proposed [2,5]. ID-based public key setting can be a good alternative for certificated-based public key setting, especially when efficient key management and moderate security are required.

In 2003, Zhang et al. [16] proposed an ID-based proxy signature scheme from pairings. However, no formal analysis was presented for the security of their scheme. In 2005, Xu et al. [15] proposed an ID-based proxy signature scheme from pairings, but the security model defined in their paper did not consider the case of an adaptively chosen identity adversary A. In other words, the target identity ID_1 is given to A before A submits queries. Later, Shim [13] proposed an identity based proxy signature scheme secure against an adaptively chosen message and chosen identity adversary. Nevertheless, we demonstrate that the above-mentioned identity based proxy signature schemes proposed in [13, 15] are vulnerable to proxy key exposure attack.

Wu et al. [14] also proposed an identity based proxy signature scheme secure against an adaptively chosen message and chosen identity adversary. As the adversary defined in [14] can be divided into three types, it is not easy to analyze the security of their scheme. Moreover, the model defined in [14] did not take proxy key exposure attack into account. Nevertheless, their scheme is secure against proxy key exposure attack.

The rest of this paper is organized as follows. At first, we develop a simplified security model for identity based proxy signature schemes. Then we propose an efficient identity based proxy signature scheme from pairings. In the following, we show that our scheme is existential unforgeable against chosen message and chosen identity attacks. In particular, the new scheme is secure against proxy key exposure attack. Finally, we compared the efficiency of our scheme with the schemes proposed in [13, 14, 15].

2. Preliminaries

2.1 Bilinear pairing

Let $< G_1, + >$ be a cyclic additive group generated by P, whose order is a large prime q, $< G_2, \bullet >$ be a cyclic multiplicative group of the same order, and let $e : G_1 \times G_1 \to G_2$ be a bilinear pairing with the following properties:

1. Bilinear: For any Q, R, $T \in G_1$, $e(Q+R,T) = e(Q,T) \bullet e(R,T)$ and $e(Q,R+T) = e(Q,R) \bullet e(Q,T)$

2. Non-degenerate: There exists $R, T \in G_1$, such that $e(R,T) \neq 1$

3. Computable: There exists an efficient algorithm to compute $e(R,T)$ for any $R, T \in G_1$.

2.2 Computational Diffie-Hellman Assumption

Computational Diffie-Hellman (CDH) problem: Let G_1 be a cyclic additive group generated by P, whose order is a prime q. Given $< P, a \cdot P, b \cdot P >$, $a, b \in Z_q$, compute $(ab) \cdot P$.

The success probability of an algorithm A in solving the CDH problem on G_1 is

$$Succ_{P,G_1}^{CDH}(A) = \Pr[A(P, a \cdot P, b \cdot P) = ab \cdot P : \forall a, b \in Z_q]$$

A (t, ε)-CDH solver A is a probabilistic polynomial-time algorithm running in time at most t such that the success probability $Succ_{P,G_1}^{CDH}(A) \geq \varepsilon$. We say that G_1 satisfies the CDH assumption if there is no polynomial time (t, ε)-CDH solver A with advantage ε non-negligible.

3. Weakness of some existing schemes

In this section, we demonstrate that the identity based proxy signature schemes proposed in [13,15] are vulnerable to proxy key exposure attack. That is, upon exposure of a proxy key, an adversary can recover the private key of the corresponding proxy.

3.1 Xu et al.'s scheme

The proxy key in Xu et al.'s scheme [15] is defined as follows:

$skp = H_4(ID_i, ID_j, m_\omega, U_\omega)d_j + V_\omega$, where m_ω is a warrant, (U_ω, V_ω) is the signature on m_ω produced by the original signer ID_i, and d_j is the secret key of the proxy signer ID_j. Since proxy signatures are often proposed for use in a potentially hostile environment, we should not assume that there is a secure channel between the original signer and the proxy. That is, (U_ω, V_ω) can be obtained by the adversary. Upon exposure of the proxy key skp, the adversary can recover d_j as follows:

$$d_j = H_4(ID_i, ID_j, m_\omega, U_\omega)^{-1}(skp - V_\omega)$$

3.2 Shim's scheme

We briefly review Shim's scheme [13] at this point.

Setup:

(1) Generates two groups G_1, G_2, two different generators $P, Q \in G_1$ and an admissible pairing $e: G_1 \times G_1 \to G_2$

(2) Picks a random number $s \in Z_q^*$ and sets the master public/secret key pair $< sP, s >$. Let $P_{pub} = sP$.

(3) Chooses three secure hash functions H_1, H_2, H_3, which are defined as follows: $H_1: \{0,1\}^* \to G_1$, $H_i: \{0,1\}^* \to Z_q$, $i = 2,3$.

Extract: Given an identity ID, computes $Q_{ID} = H_1(ID)$ and set the secret key $S_{ID} = sQ_{ID}$.

Then the original signer A produces a signature for a warrant ω as follows:

(1) A chooses $r_A \in Z_q^*$, and computes

$$U_A = r_A P \in G_1, \quad h_A = H_2(\omega, U_A) \in Z_q, V_A = h_A S_A + r_A Q \in G_1$$

After verifying the correctness of (ω, U_A, V_A), the proxy signer B computes

$h_B = H_3(\omega, U_A)$ and the proxy key is $\sigma_P = V_A + h_B S_B$, where $S_B = sQ_{ID_B}$ is the secret key of B. It is easy to see that S_B can be recovered as $S_B = (h_B)^{-1}(\sigma_P - V_A)$

4. Identity based proxy signature

4.1 Syntax of identity based proxy signature schemes

An identity based proxy signature scheme(**IBPS**) consists of the following polynomial-time algorithms:

1. MasterKeyGen(Master Key Generation): On input a security parameter $k \in \mathrm{N}$, it generates a master public/secret key pair (mpk, msk) and a list of system parameters **params**. The algorithm is assumed to be run by a Key Generation Center (KGC).

2. UserKeyGen(User Key Generation): On input msk, a user identity $ID \in \{0,1\}^*$, it generates a user secret key $Sk_{ID} \leftarrow \mathbf{UserKeyGen}\ (ID, msk)$. The algorithm is run by KGC for each user and the generated secret key is assumed to be distributed securely to the corresponding user.

3. Sign(Signature Generation): On input a user identity ID, a user secret key Sk_{ID} and a message m, it generates a standard signature $\sigma \leftarrow \mathbf{Sign}(ID, Sk_{ID}, m)$.

4. Verf(Signature Verification): On input a user identity ID, mpk, the signed message m and the standard signature σ, $\mathbf{Verf}(mpk, ID, m, \sigma)$ returns 1 if the standard signature is accepted, and 0 otherwise.

5. ProxyKeyGen(Proxy Key Generation): There is a pair of interactive algorithms (D, V) (D and V represent the original signer ID_i and the proxy signer ID_j respectively executing the proxy-designation protocol). The input to D, V includes the identities ID_i, ID_j. D also takes as input the secret key Sk_{ID_i} of the original signer, a warrant m_w consisting of the identities ID_i, ID_j, the delegation duration, the type of message delegated, etc. V also takes as input the secret key Sk_{ID_j} of the proxy signer. As a result of the

interaction, the expected local output of V is a proxy signing key $psk_{[i\to j]}$ that can be used by ID_j to produce valid proxy signatures on behalf of ID_i.

6. Proxy_Sign(Proxy Signature Generation): On input a proxy signer's identity ID_j, a proxy signing key $psk_{[i\to j]}$, the secret key Sk_{ID_j} of the proxy signer, a warrant m_w and a message m, it generates a proxy signature as follows:

$$p\sigma \leftarrow \textbf{Proxy\_Sign}(ID_j, psk_{[i\to j]}, Sk_{ID_j}, m_w, m).$$

7. Proxy_Verf(Proxy Signature Verification): On input mpk, the warrant m_w, the signed message m and the proxy signature $p\sigma$, $\textbf{Proxy\_Verf}(mpk, m_w, m, p\sigma)$ returns 1 if the proxy signature is accepted, and 0 otherwise.

8. IDP(Proxy Identification): On input a warrant m_w, and a proxy signature $p\sigma$, the proxy identification algorithm returns the identity of the designated proxy signer after verifying the correctness of the proxy signature.

Note that $<$**MasterKeyGen , UserKeyGen, Sign, Verf**$>$ can be regarded as a standard identity based signature scheme.

Correctness: We require that for all m_w, $m\in\{0,1\}^*$, $ID\in\{0,1\}^*$, $k\in \mathrm{N}$, $(mpk, msk) =$ **MasterKeyGen**(1^k), Sk_{ID} =**UserKeyGen** (ID, msk), if

(1) $psk_{[i\to j]}$ is generated by $[D(ID_i, ID_j, Sk_{ID_i}, m_w) \leftrightarrow V(ID_i, ID_j, Sk_{ID_j})]$, and

(2) $p\sigma$ =**Proxy_Sign** $(ID_j, psk_{[i\to j]}, Sk_{ID_j}, m_w, m)$, and

(3) the message m does not violate the warrant m_w.

then **Proxy_Verf** $(mpk, m_w, m, p\sigma)$ returns 1, IDP $(m_w, p\sigma) = ID_j$.

4.2 Security model

In this section, we define the security model for identity based proxy signature schemes as follows.

Let **IBPS** be an identity based proxy signature scheme, and $k\in$ **N** be a security

parameter. Define a game $Exp_{IBPS}^{A}(k)$ in which an adversary A interacts with a game challenger S_1. We use $\varnothing$ to denote an empty set.

Phase 1: S_1 runs **MasterKeyGen** (1^k) to get (mpk, msk) and a list of system parameters **params**. Then $Corr$ is initialized with $\varnothing$ which is used to keep track of the corrupted users' identities. S_1 gives mpk, **params** to A while keeping msk secret.

Phase 2: A issues the following queries:

1. CreateUser: On input an identity ID, if ID has already been created, S_1 returns a message to indicate the fact. Otherwise, the challenger executes $Sk_{ID} \leftarrow$ **UserKeyGen** (ID, msk) and an empty array $Pkey_{ID}$ is created which is used to store the proxy keys to be generated by ID. At this point, ID is said to be created.

2. RevealSecretKey: On input an identity ID, the challenger returns the corresponding user's secret key sk_{ID} if ID has been created. Then $Corr \leftarrow Corr \bigcup \{ID\}$. Otherwise a symbol $\perp$ is returned.

3. Sign_Msg: On input an identity ID and a message m adaptively chosen by A, the challenger first queries **RevealSecretKey** (ID) to get Sk_{ID} and returns a standard signature $\sigma \leftarrow$ **Sign** (ID, Sk_{ID}, m). Note that if $Sk_{ID} = \perp$, a symbol $\perp$ is returned.

4. DesignateProxy: A adaptively chooses identities ID_i (the original signer), ID_j (the proxy signer) and a warrant m_w. A requests S_1 to run the proxy-designation protocol on input (ID_i, ID_j, m_w). Then A sees the transcript of the interaction. After a successful run, the private output $psk_{[i \to j]}$ is stored in $Pkey_i[j][t]$, where t denotes the last unoccupied position of $Pkey_i[j]$. Note that A is allowed to see the computational and memory history of the corrupted identities in $Corr$.

5. Proxy_Sign_Msg: A adaptively chooses identities ID_i (the original signer), ID_j (the proxy signer), a warrant m_w, a message m, $t \in \mathrm{N}$ and requests S_1 to produce a proxy

signature. If the proxy signing key $Pkey_i[j][t]$ and Sk_{ID_j} are defined, S_1 returns $p\sigma \leftarrow \textbf{Proxy\_Sign}(ID_j, Pkey_i[j][t], Sk_{ID_j}, m_w, m)$. Otherwise a symbol $\perp$ is returned.

6. Reveal_Proxy_Key: A adaptively chooses identities ID_i (the original signer), ID_j (the proxy signer), $t \in \mathrm{N}$. If the proxy signing key $Pkey_i[j][t]$ is defined, S_1 returns $Pkey_i[j][t]$. Otherwise a symbol $\perp$ is returned.

Phase 3: A wins if one of the following events happens:

(1) A outputs (ID^*, m^*, σ^*), such that $\textbf{Verf}(mpk, ID^*, m^*, \sigma^*) = 1$. We require that A never made a **Sign_Msg** query on (ID^*, m^*), nor $ID^* \in Corr$ (forgery of a standard signature).

(2) A outputs $(ID_i, ID_j, m_w^*, m^*, p\sigma^*)$ after making a **DesignateProxy** request on (ID_i, ID_j, m_w^*), such that $\textbf{Proxy\_Verf}(mpk, m_w^*, m^*, p\sigma^*) = 1$, $\mathrm{IDP}(m_w^*, p\sigma^*) = ID_j$. We require that A never made a **Proxy_Sign_Msg** query on $(ID_i, ID_j, t, m_w^*, m^*)$, for some $t \in \mathrm{N}$, nor $ID_j \in Corr$. However, A is allowed make a **RevealSecretKey** query on ID_i (forgery of a proxy signature by ID_j on behalf of ID_i ; ID_j has been designated by ID_i). This case simulates attacks when the adversary is able to compromise the secret key of the original signer.

(3) A outputs $(ID_i, ID_j, m_w^*, m^*, p\sigma^*)$ without making a **DesignateProxy** request on (ID_i, ID_j, m_w^*), such that $\textbf{Proxy\_Verf}(mpk, m_w^*, m^*, p\sigma^*) = 1$, $\mathrm{IDP}(m_w^*, p\sigma^*) = ID_j$. We require that $|\{ID_i, ID_j\} \cap Corr| \leq 1$ (forgery of a proxy signature by ID_j on behalf of ID_i ; ID_j is not designated by ID_i). This case simulates attacks when the adversary tries to produce a proxy signature without running the proxy-designation protocol.

Finally, S_1 returns 1 to indicate the adversary‘s success. We define the success

probability of the adversary as $Succ_{IBPS}(A) = \Pr[Exp_{IBPS}^{A}(k) = 1]$.

An identity based proxy signature scheme is existential unforgeable against chosen message and chosen identity attacks if for any probabilistic polynomial time (**PPT**) adversary A, the success probability $Succ_{IBPS}(A)$ is negligible.

The adversary defined in [14] can be divided into three types:

Type I: This type of adversary only has the identities of the original signer and the proxy signer.

Type II: This type of adversary has the identities of the original signer and the proxy signer, and also can have the secret key of the proxy signer.

Type III: This type of adversary has the identities of the original signer and the proxy signer, and also can have the secret key of the original signer.

It is obvious that the adversary defined in our model can simulate all the attacks captured by the model defined in [14]. For example, Type I attackers who know only some target identities can be modeled by our attacker who does not issue any **RevealSecretKey** query. In addition, the model defined in [14] did not take proxy key exposure attack into account. Finally, our simplified model may lead to simpler proofs of security.

5. Our scheme

In this section, we propose an efficient identity based proxy signature scheme from bilinear pairing. The proposed scheme consists of the following algorithms:

MasterKeyGen: Assume k is the security parameter of our system. Let $<G_1, +>$ be a cyclic additive group generated by P, whose order is a large prime q, $<G_2, \bullet>$ be a cyclic multiplicative group of the same order, and let $e: G_1 \times G_1 \to G_2$ be a bilinear map. Then KGC performs the following operations:

(1) Picks a random number $s \in Z_q^*$ and sets the master public/secret key pair $<mpk, msk> = <s \cdot P, s>$.

(2) Chooses three secure one-way functions H_1, H_2, H_3, which are defined as follows:

$H_1:\{0,1\}^* \to G_1$, $H_2:\{0,1\}^* \times G_1 \to G_1$, $H_3:\{0,1\}^* \times G_1 \to G_1$.

(3) Sets the system parameters **params** as $<(G_1,+),(G_2,\bullet),e,q,P,mpk,H_1,H_2,H_3>$.

UserKeyGen: On input an identity ID_i, KGC computes $Q_i = H_1(ID_i)$, $Sk_i = msk \cdot Q_i$. Then KGC distributes Sk_i to the corresponding user identified by ID_i as his secret key over a secure channel. The user can verify the correctness by checking $e(Sk_i, P) = e(Q_i, mpk)$.

Sign: In order to sign a message m, the user identified by ID_i should perform the following steps:

(1) Picks a random number $k_i \in Z_q^*$ and computes $K_i = k_i \cdot P$.

(2) Computes $V_i = H_2(ID_i, m, K_i)$, $U_i = k_i \cdot V_i + Sk_i$.

(3) The signature is $\sigma = <U_i, K_i>$.

Verf: Given the master public key mpk, an identity ID_i, the signed message m, the correctness of the standard signature σ can be verified as follows:

(1) Computes $V_i = H_2(ID_i, m, K_i)$.

(2) Returns 1 if and only if $e(U_i, P) = e(V_i, K_i) \bullet e(Q_i, mpk)$, where $Q_i = H_1(ID_i)$.

It is trivial to check the correctness of the verification equation.

ProxyKeyGen: In order to delegate the signing capability to a proxy signer ID_j, the original signer ID_i should generate a warrant m_w consisting of the identities of the original signer and the proxy signer, the delegation duration, the type of message delegated, etc. The proxy-designation protocol can be described as follows:

(1) The original signer ID_i outputs $\sigma' = \mathbf{Sign}(ID_i, Sk_i, m_w)$, where $\sigma' = <U_i', K_i'>$.

Then $<\sigma', m_w>$ is sent to the proxy signer ID_j.

(2) If $\mathbf{Verf}(mpk, ID_i, m_w, \sigma') = 1$, ID_j proceeds to the next step. Otherwise ID_j requests ID_i to provide a valid signature for the warrant m_w.

(3) The proxy signing key of ID_j is $psk_{[i\to j]}=<U_i^{/},K_i^{/}>$.

Proxy_Sign: Given the proxy signing key $psk_{[i\to j]}=<U_i^{/},K_i^{/}>$, in order to generate a proxy signature for a message m on behalf of ID_i, ID_j should perform the following steps:

(1) Picks a random number $k_j \in Z_q^*$ and computes $K_j = k_j \cdot P$.

(2) Computes $V_j = H_3(ID_i, ID_j, m_w, m, (K_j + K_i^{/}))$.

(3) Computes $U_j = U_i^{/} + Sk_j + k_j \cdot V_j$

The proxy signature is $p\sigma = <U_j, K_j, K_i^{/}>$.

Proxy_Verf: After receiving the signed message m, the warrant m_w and the proxy signature $p\sigma = <U_j, K_j, K_i^{/}>$, a verifier should perform the following steps:

(1) If the warrant m_w is invalid, (for instance, the time period for delegation expired), $p\sigma$ is rejected. Otherwise the verifier extracts ID_i, ID_j from m_w and proceeds to the next step.

(2) Computes $V_j = H_3(ID_i, ID_j, m_w, m, (K_j + K_i^{/}))$, $V_i^{/} = H_2(ID_i, m_w, K_i^{/})$.

(3) Returns 1 if and only if

$$e(U_j, P) = e(V_i^{/}, K_i^{/}) \cdot e(V_j, K_j) \cdot e(Q_i + Q_j, mpk)$$

The correctness of the verification equation can be verified as follows:

$$\begin{aligned} e(U_j, P) &= e(U_i^{/}, P) \cdot e(Sk_j, P) \cdot e(k_j \cdot V_j, P) \\ &= e(V_i^{/}, K_i^{/}) \cdot e(V_j, K_j) \cdot e(Q_i, mpk) \cdot e(Q_j, mpk) \\ &= e(V_i^{/}, K_i^{/}) \cdot e(V_j, K_j) \cdot e(Q_i + Q_j, mpk) \end{aligned}$$

IDP: On input a warrant m_w, and a proxy signature $p\sigma$, the proxy identification algorithm returns the identity of the designated proxy signer contained in m_w after verifying the correctness of the proxy signature $p\sigma$.

6.Security analysis

6.1 Security Proof

Suppose a polynomial-time adversary A can break our **IBPS** scheme with non-negligible success probability ε in time at most t. We show how to construct a PPT algorithm B that uses A to solve the CDH problem on G_1 with non-negligible probability by using techniques from [3].

Let $(X = a \cdot P, Y = b \cdot P) \in G_1 \times G_1$, $a, b \in Z_q$, be an instance of the CDH problem taken as input by B. Then B works by interacting with the adversary A (B simulates the game challenger).

The system parameters **params** are $< (G_1, +), (G_2, \bullet), e, q, P, mpk, H_1, H_2, H_3 >$, where mpk is initialized with X and H_1, H_2, H_3 are random oracles controlled by B.

During the simulation, B can answer A's queries as follows:

H_1 **Queries:** B maintains a list $H_1^{list} = \{< ID, Q_{ID}, l, coin >\}$, where $coin \in \{0,1\}$. If the queried identity ID appears on the H_1^{list} in a tuple $\{< ID, Q_{ID}, l, coin >\}$, B responds to A with $H_1(ID) = Q_{ID}$. Otherwise B picks a random $l \in Z_q^*$, sets $(Q_{ID} = l \cdot P,\ coin = 0)$ with probability δ, or $(Q_{ID} = l \cdot Y,\ coin = 1)$ with probability $1 - \delta$. Then B adds $< ID, Q_{ID}, l, coin >$ to the H_1^{list} and responds to A with $H_1(ID) = Q_{ID}$.

CreateUser: B maintains a list $L = \{< ID, Sk_{ID} >\}$. Suppose the query is made on an identity ID. B performs as follows:

(1) If the list L contains a tuple $< ID, Sk_{ID} >$, B returns a message to indicate the fact.

(2) Otherwise, B queries $H_1(ID)$. Then B looks up the H_1^{list} to extract a tuple $< ID, Q_{ID}, l, coin >$. If $coin$=1, B adds $< ID, \perp >$ to the list L. If $coin$=0, B computes $Sk_{ID} = l \cdot mpk$, adds $< ID, Sk_{ID} >$ to the list L, and creates an empty

array $Pkey_{ID}$ which is used to store the proxy keys to be generated by ID.

RevealSecretKey: Suppose the query is made on an identity ID. At first, B looks up the list L. If L contains a tuple $< ID, Sk_{ID} >$ and $Sk_{ID} \neq \perp$, B returns Sk_{ID}. Otherwise, B returns $\perp$ and aborts.

H_2 **Queries:** B maintains a list H_2^{list}. Suppose the query is made on $< ID, m, K >$. If a tuple $< ID, m, K, h_{ID}^{(2)} >$ is already in the H_2^{list}, B returns $h_{ID}^{(2)} \cdot P$. Otherwise B picks a random $h_{ID}^{(2)} \in Z_q^*$, and responds to A with $H_2(ID, m, K) = h_{ID}^{(2)} \cdot P$. Then $< ID, m, K, h_{ID}^{(2)} >$ is added to the H_2^{list}.

Sign_Msg: On input $< ID, m >$. B should perform as follows:

(1) Executes $Sk_{ID} \leftarrow$ **RevealSecretKey** (ID). If $Sk_{ID} = \perp$, B returns $\perp$ and aborts.

(2) Otherwise B picks a random number $k \in Z_q^*$ and computes $K = k \cdot P$.

(3) Executes $V \leftarrow H_2(ID, m, K)$ and computes $U = k \cdot V + Sk_{ID}$.

(4) B returns $\sigma = < U, K >$.

DesignateProxy: Suppose A makes a query on ID_i(the original signer), ID_j (the proxy signer), and m_w. If $Sk_{ID_i} = \perp$, B returns $\perp$ and aborts. Otherwise, B queries the oracle **Sign_Msg** with (ID_i, m_w). If a valid signature $\sigma' = < U_i', K_i' >$ is generated, B forwards σ' to A. Finally, B sets $Pkey_i[j][t] \leftarrow \sigma'$, where t denotes the last unoccupied position of $Pkey_i[j]$. Obviously, as the proxy signing key is just the signature on the warrant, it is useless to provide the adversary with a **Reveal_Proxy_Key** oracle in this case.

H_3 **Queries:** B maintains a list H_3^{list}. Suppose the query is made on $< ID_i, ID_j, m_w, m, K_{i,j} >$. If a tuple $< ID_i, ID_j, m_w, m, K_{i,j}, h_{i,j}^{(3)} >$ is already in the H_3^{list}, B returns $h_{i,j}^{(3)} \cdot P$. Otherwise, B picks a random $h_{i,j}^{(3)} \in Z_q^*$, and responds to A with

$H_3(ID_i, ID_j, m_w, m, K_{i,j}) = h_{i,j}^{(3)} \cdot P$. Then $< ID_i, ID_j, m_w, m, K_{i,j}, h_{i,j}^{(3)} >$ is added to the H_3^{list}.

Proxy_Sign_Msg: Without loss of generality, we assume that A always makes a query on $< ID_i, ID_j, t, m_w, m >$ after making a successful **DesignateProxy** query on $< ID_i, ID_j, m_w >$. Then B performs as follows:

(1) If $Pkey_i[j][t] = \sigma^{/}$ ($\sigma^{/} = < U_i^{/}, K_i^{/} >$) is not defined, B returns $\perp$. Otherwise B proceeds to the next step.

(2) B executes $Sk_{ID_j} \leftarrow$ **RevealSecretKey** (ID_j). If $Sk_{ID_j} = \perp$, B returns $\perp$ and aborts.

(3) Picks a random number $k_j \in Z_q^*$ and computes $K_j = k_j \cdot P$.

(4) Executes $V_j \leftarrow H_3(ID_i, ID_j, m_w, m, (K_j + K_i^{/}))$.

(5) Computes $U_j = U_i^{/} + Sk_{ID_j} + k_j \cdot V_j$

(6) B returns $p\sigma = < U_j, K_j, K_i^{/} >$.

Eventually, A halts and outputs a successful forgery (In order to be successful, the restrictions on A defined in our model must be satisfied). We should consider the following cases:

(1) Suppose A outputs a forgery of the form $\sigma^* = < U^*, K^* >$ on a message m^* for a created identity ID^*. Then B looks up the H_1^{list} to extract a tuple $< ID^*, Q_{ID^*}, l, coin >$.

If $coin$=0, B reports failure and terminates. If $coin$=1, B looks up the H_2^{list} to find out a tuple $< ID^*, m^*, K^*, h_{ID^*}^{(2)} >$. Since the forgery is successful, the probability of the event that A does not make a H_2 query on $< ID^*, m^*, K^* >$ is at most $1/q$, which is negligible. Hence we know that $H_2(ID^*, m^*, K^*) = h_{ID^*}^{(2)} \cdot P$ with probability at least $1 - 1/q$. So we have

$e(U^*, P) = e(V^*, K^*) \bullet e(Q_{ID^*}, mpk)$, where $V^* = h_{ID^*}^{(2)} \cdot P$

$= e(h_{ID^*}^{(2)} \cdot P, K^*) \bullet e(l \cdot Y, X)$

Hence $e(Y, X)^l = e(U^* - h_{ID^*}^2 \cdot K^*, P)$.

Then B outputs $l^{-1} \cdot (U^* - h_{ID^*}^2 \cdot K^*)$ as the solution to the given instance of the CDH problem on G_1.

(2) Suppose A outputs a forgery $< ID_i, ID_j, m_w, U_j, K_j, K_i^{/} >$ on a message m^* after making a **DesignateProxy** query on (ID_i, ID_j, m_w). Then B looks up the H_1^{list} to extract $< ID_i, Q_{ID_i}, l_i, coin_i >$, $< ID_j, Q_{ID_j}, l_j, coin_j >$.

If $coin_i = 0 \wedge coin_j = 0$, B reports failure and terminates. Otherwise, B looks up the H_2^{list}, H_3^{list} to extract $< ID_i, m_w, K_i^{/}, h_{ID_i}^{(2)} >$, $< ID_i, ID_j, m_w, m, K_i^{/} + K_j, h_{i,j}^{(3)} >$ respectively. Since the forgery is successful, the probability of the event that A does not query H_2 or H_3 is at most $2/q$. Hence we know that these tuples are already in the H_2^{list} and H_3^{list} with probability at least $1 - 2/q$. So we have

$e(U_j, P) = e(V_i^{/}, K_i^{/}) \bullet e(V_j, K_j) \bullet e(Q_{ID_i}, mpk) \bullet e(Q_{ID_j}, mpk)$

where $V_j \leftarrow H_3(ID_i, ID_j, m_w, m, (K_j + K_i^{/}))$, $V_i^{/} \leftarrow H_2(ID_i, m_w, K_i^{/})$

Then consider the following sub-cases:

(2.1) $coin_i = 1 \wedge coin_j = 0$.

$e(U_j, P) = e(h_{ID_i}^{(2)} \cdot P, K_i^{/}) \bullet e(h_{i,j}^{(3)} \cdot P, K_j) \bullet e(l_i \cdot Y, X) \bullet e(l_j \cdot P, X)$

Hence $e(Y, X)^{l_i} = e(U_j - h_{ID_i}^{(2)} \cdot K_i^{/} - h_{i,j}^{(3)} \cdot K_j - l_j \cdot X, P)$.

Then B outputs $(l_i)^{-1} \cdot (U_j - h_{ID_i}^{(2)} \cdot K_i^{/} - h_{i,j}^{(3)} \cdot K_j - l_j \cdot X)$ as the solution to the given instance of the CDH problem on G_1.

(2.2) $coin_i = 0 \wedge coin_j = 1$.

$e(U_j, P) = e(h_{ID_i}^{(2)} \cdot P, K_i^{/}) \bullet e(h_{i,j}^{(3)} \cdot P, K_j) \bullet e(l_i \cdot P, X) \bullet e(l_j \cdot Y, X)$

Hence $e(Y, X)^{l_j} = e(U_j - h_{ID_i}^{(2)} \cdot K_i^{/} - h_{i,j}^{(3)} \cdot K_j - l_i \cdot X, P)$.

Then B outputs $(l_j)^{-1} \cdot (U_j - h_{ID_i}^{(2)} \cdot K_i^{/} - h_{i,j}^{(3)} \cdot K_j - l_i \cdot X)$ as the solution to the given instance of the CDH problem on G_1.

(2.3) $coin_i = 1 \wedge coin_j = 1$.

$$e(U_j, P) = e(h_{ID_i}^{(2)} \cdot P, K_i^{/}) \bullet e(h_{i,j}^{(3)} \cdot P, K_j) \bullet e(l_i \cdot Y, X) \bullet e(l_j \cdot Y, X)$$

Hence $e(Y, X)^{l_i + l_j} = e(U_j - h_{ID_i}^{(2)} \cdot K_i^{/} - h_{i,j}^{(3)} \cdot K_j, P)$.

Then B outputs $(l_i + l_j)^{-1} \cdot (U_j - h_{ID_i}^{(2)} \cdot K_i^{/} - h_{i,j}^{(3)} \cdot K_j)$ as the solution to the given instance of the CDH problem on G_1.

(3) Suppose A outputs a forgery $< ID_i, ID_j, m_w, U_j, K_j, K_i^{/} >$ on a message m^* without making a **DesignateProxy** request on (ID_i, ID_j, m_w). The analysis of this case is similar to that of case (2).

Claim 1: If the algorithm B does not abort during the simulation, then the view of the adversary A in the simulated game is indistinguishable from that in the real game.

Proof: At first, the responses to H_1, H_2, H_3 queries are as in the real game since each response is uniformly distributed over G_1. If the algorithm B does not abort, the responses to **RevealSecretKey, Sign_Msg**, **DesignateProxy**, **Proxy_Sign_Msg** queries are valid. Hence the view of the adversary A in the simulated game in this case is indistinguishable from that in the real game.

Lemma 1: Assume G_1 satisfies the CDH assumption. Suppose there is a polynomial-time adversary A can existentially forge a standard signature of our **IBPS** scheme with success probability ε in time at most t. Suppose A makes at most q_c **CreateUser** queries **,** q_{H_i} queries to random oracles H_i for i =1,2, q_{sig} **Sign_Msg** queries and q_{rev} **RevealSecretKey** queries . Then there is an algorithm B that solves the CDH problem on

G_1 with probability

$$\varepsilon' \approx \frac{\varepsilon}{q_{sig}+q_c}(1-\frac{1}{q_{sig}+q_c+1})^{q_{sig}+q_c+1}$$

Proof: The probability that B does not abort in this case (i.e., B can answer all standard signature queries and **RevealSecretKey** queries) is at least $\delta^{q_{sig}+q_{rev}}$. The reason is that if B is able to obtain the secret key of a user, he can answer signature queries with regard to that user perfectly. When B answers **CreateUser** queries, it is easy to show that with probability δ he can generate the secret key of a user correctly. Hence, B can answer one signature query or **RevealSecretKey** query correctly with probability δ.

Then B outputs the solution to the instance of the CDH problem with probability $(1-\delta)(1-1/q)$. Hence B is able to solve the CDH problem with success probability at least $\varepsilon\ (1-\delta)(1-1/q)\ \delta^{q_{sig}+q_{rev}} \approx \varepsilon\ (1-\delta)\ \delta^{q_{sig}+q_{rev}}$.

Let $\lambda = \varepsilon\ (1-\delta)\ \delta^{q_{sig}+q_{rev}}$, $a = q_{sig}+q_{rev}$. By an analysis similar to Coron's techniques [3], the success probability λ is maximized at $\delta_{opt} = \frac{a}{a+1}$. Hence the success probability ε' of $B \approx \frac{\varepsilon}{a}(1-\frac{1}{a+1})^{a+1}$, and for large a, $\varepsilon' \approx \frac{\varepsilon}{\exp(1)\cdot a}$.

The running time of B can be calculated as

$$t + (q_c + q_{H_1} + q_{H_2} + q_{sig})t_m + q_{rev}q_c O(1)$$

where t_m is the time to compute a scalar multiplication in G_1.

Lemma 2: Assume G_1 satisfies the CDH assumption. Suppose there is a polynomial-time adversary A can existentially forge a proxy signature of our **IBPS** scheme with success probability ε in time at most t (Here we only consider case (2) of our security model for the sake of simplicity). Suppose A makes at most q_c **CreateUser** queries **,** q_{H_i} queries to random oracles H_i for i =1,2,3, q_{sig} **Sign_Msg** queries, q_{psig} **Proxy_Sign_Msg** queries, q_{desg} **DesignateProxy** queries and q_{rev} **RevealSecretKey** queries . Then there is

an algorithm B that the CDH problem on G_1 with probability

$$\varepsilon' \approx \frac{\varepsilon}{b}(1-\frac{1}{1+b})^{1+b}, \text{ where } b=\frac{(q_{sig}+q_{desg}+q_{psig}+q_{rev})}{2}$$

Proof: The probability that B does not abort in this case (i.e., B can answer all standard signature queries, proxy-designation queries, proxy signature queries and **RevealSecretKey** queries is at least $\delta^{q_{sig}+q_{desg}+q_{psig}+q_{rev}}$. The reason is that if B obtains the secret key of the proxy signer, B can respond to a proxy signature query successfully as we assume A always makes a proxy signature query after making a successful proxy-designation query. Hence, B can answer one proxy signature query successfully with probability δ.

Then B outputs the solution to the CDH problem with probability $(1-\delta^2)(1-2/q)$ (except for the sub-case $coin_i=0 \wedge coin_j=0$). Hence B is able to solve the CDH problem with the following probability:

$$\varepsilon\ (1-\delta^2)(1-2/q)\ \delta^{q_{sig}+q_{desg}+q_{psig}+q_{rev}} \approx \varepsilon\ (1-\delta^2)\ \delta^{q_{sig}+q_{desg}+q_{psig}+q_{rev}}.$$

Let $\lambda = \varepsilon\ (1-\delta^2)\ \delta^{q_{sig}+q_{desg}+q_{psig}+q_{rev}}$, $b = \frac{(q_{sig}+q_{desg}+q_{psig}+q_{rev})}{2}$. The success probability λ is maximized at $\delta_{opt}=\sqrt{\frac{b}{b+1}}$. Hence the success probability ε' of B

$\approx \frac{\varepsilon}{b}(1-\frac{1}{1+b})^{1+b}$, and for large b, $\varepsilon' \approx \frac{\varepsilon}{\exp(1)\cdot b}$

The running time of B can be calculated as

$$t+(q_c+q_{H_1}+q_{H_2}+q_{H_3}+q_{sig}+q_{psig}+q_{desg})t_m + q_{rev}q_cO(1)$$

where t_m is the time to compute a scalar multiplication in G_1.

Case (3) of our security model can be analyzed similarly.

6.2 Discussion

Verifiability: It can be derived from the correctness of the proxy signature verification equation discussed in section 4.

Unforgeability: It is established from the conclusion of Lemma 2.

Strong identifiability: The identity of the proxy signer can be extracted from the

warrant after verifying the correctness of the proxy signature.

Undeniability: It can be derived from unforgeability and strong identifiability.

Prevention of misuse: The proxy signer can only produce proxy signatures without violating the warrant m_w signed by the original signer. Moreover, the proxy signing key is just the standard signature on the warrant m_w. According to Lemma 1, the standard signature scheme defined in our scheme is unforgeable. Hence the proxy signer cannot sign messages that have not been authorized by the original signer.

Proxy key exposure: As the proxy signing key is just the signature on the warrant, it is harmless to provide the proxy signing key with the adversary in this case. Hence our scheme is secure against proxy key exposure attack.

7. Performance Analysis

In this section, we evaluate the performance of the proposed scheme and other related schemes proposed in [13, 14, 15] in terms of the signature length and computational cost. Let $|G_1|$ be the bit length of an element in G_1. Mu and Ad denote scalar multiplication and addition in G_1 respectively. H denotes a hash operation . Exp and P denote an exponentiation operation in G_2 and a pairing operation respectively, which are the most time-consuming operation. The result is stated in Table 1. It is easy to check the correctness of the numbers listed in Table 1. Obviously, the schemes proposed in [13, 15] is vulnerable to proxy key exposure attack. In contrast to Wu et al.'s scheme [14], our scheme is more efficient in terms of the computational cost. Moreover, the model defined in [14] did not take proxy key exposure attack into account. Finally, our simplified model may lead to simpler proofs of security.

8.Conclusion

In this paper, we focus on realizing an identity based proxy signature scheme in order to combine the advantages of these concepts. At first, a simplified security model for identity based proxy signature scheme is established. Then an efficient identity based proxy signature

scheme from bilinear pairing is proposed. In the following, we provide a reductionist proof to show that security of our scheme relies on the CDH problem. Finally, the performance analysis shows that in contrast to Wu et al.'s scheme [16], our scheme is more efficient in terms of the computational cost. In particular, the new scheme is secure against proxy key exposure attack.

References

[1] A.Boldyreva, A. Palacio, B. Warinschi, "Secure proxy signature schemes for delegation of signing rights", http://eprint.iacr.org/2003/096

[2] D.Boneh, M. Franklin, "Identity based encryption from the Weil pairing ", in Advances in Cryptology- Crypto'2001, LNCS 2139, Springer-Verlag, pp.213-229,2001

[3] J.-S., Coron, "On the exact security of full domain hash", in Advance in Cryptology-Crypto'2000, LNCS 1880, Springer-Verlag, pp.229-235, 2000

[4] S. Goldwasser, S. Micali, R. Rivest, , "A digital signature scheme secure against existential adaptive chosen-message attacks", SIAM Journal on Computing, 17(2), pp.281-308, 1988

[5] F. Hess, "Efficient identity based signature schemes based on pairings", in Selected Areas in Cryptography-SAC'2002, LNCS 2595, Springer-Verlag, pp.310-324, 2002

[6] C.L. Hsu, T.S. Wu, T.C. Wu, "New nonrepudiable threshold proxy signature scheme with known signers", The Journal of System and Software, 58, pp.119-124, 2001

[7] S.J. Hwang, C.C. Chen, "A new proxy multi-signature scheme", in Proceedings of International Workshop on Cryptology and Network Security, pp.134-138, 2000

[8] S. Kim, S. Park, D. Won, "Proxy signature, revisited", in Proceedings of SCIS'2001, International Conference on Information and Communication Security, pp.223-232, 1997

[9] B. Lee, H. Kim, K. Kim, "Strong proxy signature and its applications", in Proceedings of ICICS'97, International Conference on Information and Communication Security, pp.603-608, 2001

[10] M. Mambo, K. Usuda, E. Okamoto, "Proxy signatures for delegating signing operation", in Proceedings of 3rd ACM Conference on Computer and Communications Security , ACM Press, pp.48-57, 1996

[11] A. Shamir, “Identity based cryptosystems and signature schemes”, in Advances in Cryptology-Crypto’84, LNCS 196, Springer-Verlag, pp.47-53,1984

[12] Jacob C.N. Schuldt, K. Matsuura, and Kennth G. Paterson, “ Proxy Signatures secure against proxy key exposure“, PKC 2008, LNCS 4939, pp.141-161, 2008

[13] K.-A. Shim, "An Identity-Based Proxy Signature Scheme from Pairings", ICICS 2006, LNCS 4307, pp. 60-71, 2006.

[14] W. Wu, Y. Mu, W. Susilo, J. Seberry, X.Y. Huang, "Identity-based Proxy Signature from Pairing", ATC 2007, LNCS 4610, Berlen, Heidelberg: Springer-Verlag, 2007. pp. 22-31.

[15] J. Xu, Z.F. Zhang, D.G. Feng, "ID-based Proxy Signature using Bilinear Pairing", ISPA 2005 Workshops, LNCS 3759, Berlen, Heidelberg: Springer-Verlag, 2005. pp. 359-367.

[16] Z.F. Zhang, K. Kim, “Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairings", ACISP 2003, LNCS 2727, pp. 312-323, 2003.

Table 1.Performance comparison with other related schemes

Scheme	Proxy signature length	Proxy signing cost	Proxy signature verification cost	Secure against Proxy key exposure
Xu et al.’s scheme [15]	$3\|G_1\|$	$2Mu+1Ad+1H$	$5P+Exp$	No
Shim’s scheme [13]	$3\|G_1\|$	$3Mu+1Ad+1H$	$3P$	No
Wu et al.’s scheme [14]	$3\|G_1\|$	$4Mu+3Ad+2H$	$5P$	Yes
Our scheme	$3\|G_1\|$	$2Mu+3Ad+1H$	$4P$	Yes