Efficient and Secure Many-to-One Signature Delegation

Rajeev Anand Sahu[1(✉)] and Vishal Saraswat[1]

C.R.Rao Advanced Institute of Mathematics Statistics and Computer Science, Hyderabad, India
rajeevs.crypto@gmail.com

Abstract. We propose an IBPMS scheme from pairings, which is more efficient in the sense of computation and operation time than the existing schemes. We also prove on random oracle that the propose d scheme is secure against existential forgery on adaptive chosen-message and adaptive-chosen ID attack under the k-CAA assumption. Additionally, our scheme fulfills all the security requirements of a proxy signature scheme. Moreover we do an efficiency analysis and show that our scheme is significantly more efficient than the existing IBPMS schemes in the sense of computation and operation time.

Keywords: Identity-based cryptography · Digital signature · Bilinear pairings · Delegation of signing rights · Proxy multi-signature · k-CAA problem · Provable security

1 Introduction

In a proxy signature scheme, an original signer o can transfer its signing rights to a proxy signer ρ without transferring its private key; and the proxy signer can sign any document on behalf of the original signer. Proxy signature schemes are applicable in distributed systems, grid computing, mobile agent environment etc. where delegation of rights is quite common.

The notion of proxy signature has been around since 1989 due to Gasser *et al.* [3] but the first formal construction of a proxy signature scheme [6] was proposed in 1996. The notion of proxy multi-signature was introduced by Yi et al. [10] in 2000 and then in 2005, Li and Chen [5] proposed the first proxy multi-signature scheme in ID-based setting using bilinear pairings. Since then, many identity (ID)-based proxy multi-signature (IBPMS) schemes have been proposed using bilinear pairings, but most of the schemes are either too much inefficient or insecure, hence cannot be considered for the practical implementation.

We propose here an IBPMS scheme from bilinear pairings. Our scheme is significantly more efficient than the existing IBPMS schemes [1,5,8,9] in the sense of computation and operation time. Moreover, we prove the security of our scheme against existential forgery on adaptive chosen-message and adaptive chosen-ID attacks in random oracle model. Additionally, we also show that the proposed scheme fulfills all the security requirements of a proxy signature scheme listed in [4].

S. Qing et al. (Eds.): ICICS 2015, LNCS 9543, pp. 252–259, 2016.
DOI: 10.1007/978-3-319-29814-6_20

2 Preliminaries

Definition 1. Let G_1 be an additive cyclic group with generator P and G_2 be a multiplicative cyclic group with generator g. Let both the groups are of the same prime order q. A map $e : G_1 \times G_1 \rightarrow G_2$ is called a *cryptographic bilinear map* or a *pairing* if it satisfies the following properties:

1. *Bilinearity*: For all $a, b \in \mathbb{Z}_q^*$, $e(aP, bP) = e(P, P)^{ab}$, or equivalently, for all $Q, R, S \in G_1$, $e(Q+R, S) = e(Q, S)e(R, S)$ and $e(Q, R+S) = e(Q, R)e(Q, S)$.
2. *Non-degeneracy*: There exists $Q, R \in G_1$ such that $e(Q, R) \neq 1$. Note that since G_1 and G_2 are groups of prime order, this condition is equivalent to the condition $e(P, P) \neq 1$, which again is equivalent to the condition that $e(P, P)$ is a generator of G_2.
3. *Computability*: There exists an efficient algorithm to compute $e(Q, R) \in G_2$, for any $Q, R \in G_1$.

Definition 2. The *k-CAA Problem* [7] is to compute $\frac{1}{s+e_0}P$, for some $e_0 \in Z_q^*$ when given $P, sP \in G_1, e_1, e_2, \ldots, e_k \in Z_q^*$ and $\frac{1}{s+e_1}P, \frac{1}{s+e_2}P, \ldots, \frac{1}{s+e_k}P \in G_1$.

Definition 3. The (t, ϵ) *k-CAA assumption* holds in G_1 if there is no algorithm which takes at most t running time and can solve the k-CAA problem with at least a non-negligible advantage ϵ.

3 Proposed IBPMS Scheme

3.1 Setup

Given a security parameter 1^K, the private key generator (PKG) generates the system's master secret $s \in \mathbb{Z}_q^*$ and the system's public parameters

$$params = (K, q, G_1, G_2, e, H_1, H_2, P, Q),$$

where G_1 is an additive cyclic group of prime order q; G_2 is a multiplicative cyclic group of prime order q; $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map defined as above; $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \times G_1 \rightarrow \mathbb{Z}_q^*$ are two cryptographic hash functions; P is a generator of G_1; and $Q := sP \in G_1$ is system's public key.

3.2 Extraction

Given a user's identity ID, the PKG computes its

- public key as: $Q_{\mathrm{ID}} = H_1(\mathrm{ID})$ and
- private key as: $S_{\mathrm{ID}} = \frac{1}{s+H_1(\mathrm{ID})}P$.

Thus, the proxy signer A_0 and the original signers, A_i, $i = 1, \ldots, n$, have their public keys and the corresponding private keys

- $Q_{\mathrm{ID}_{A_i}} = H_1(\mathrm{ID}_{A_i})$ and
- $S_{\mathrm{ID}_{A_i}} = \frac{1}{s+H_1(\mathrm{ID}_{A_i})}P$,

for $i = 0, 1, \ldots, n$.

3.3 Proxy Key Generation

In this phase, all the original signers interact with the proxy signer to delegate their signing rights through a signed warrant. The warrant w includes some specific information about the message like nature of the message, time of delegation, identity information of the original signers and the proxy signer, period of validity, some public keys etc. After the successful interaction, the proxy signer outputs its proxy signing key. The interaction and proxy key generation can be described in the following phases:

Delegation generation:

Each of the n original signers, A_i, $i = 1, \ldots, n$, and the proxy signer, A_0, interact and do the following:

- set $I = \sum_{i=0}^{n} H_1(\mathrm{ID}_{A_i}) \in \mathbb{Z}_q^*$ and $J = (n+1)Q + IP \in G_1$;
- select $x_i \overset{\$}{\leftarrow} Z_q^*$;
- compute and publish $V_i = e(J, S_{\mathrm{ID}_{A_i}}) \in G_2$, and $W_i = V_i^{x_i} \in G_2$;
- create a warrant w which includes the identities ID_{A_i} of the proxy and original signers, the values V_i and $V_o = \prod_{i=0}^{n} V_i$, the values W_i and $W_o = \prod_{i=0}^{n} W_i$, the scope of messages to be signed, time of delegation, period of validity etc.;
- computes $h_o = H_2(w, W_o) \in \mathbb{Z}_q^*$;
- and $S_i = (x_i + h_o) S_{\mathrm{ID}_{A_i}}$.
- Finally each original signer sends (w, S_i), $i = 1, \ldots, n$, to the proxy signer as a partial delegation.

Delegation verification:

On receiving (w, S_i) from each original signer A_i, the proxy signer A_0 obtains W_i and W_o from the warrant, computes $h_o = H_2(w, W_o) \in \mathbb{Z}_q^*$ and validates each partial delegation by checking

$$e(J, S_i) = W_i V_i^{h_o}.$$

If the above equality does not hold for any $1 \leq i \leq n$, the proxy signer terminates the protocol.

Proxy key generation:

In this phase, the proxy signer computes its proxy secret key to sign the message on behalf of the group of original signers to be

$$S_{pk} = \sum_{i=0}^{n} S_i.$$

3.4 Proxy Multi-signature

To sign a message $m \in \{0,1\}^*$ under the warrant w on behalf of the group of original signers, the proxy signer does the following:

- selects $y \overset{\$}{\leftarrow} Z_q^*$;
- computes $V_\rho = V_o^y$ and $W_\rho = W_o^y$;
- computes $h_\rho = H_2(m, W_\rho) \in Z_q^*$
- and $\sigma = (y + h_\rho) S_{pk}$.
- Finally, (σ, V_ρ, W_ρ) is the IBPMS by the proxy signer on behalf of the group of original signers on message m under the warrant w.

3.5 Verification:

On receiving the IBPMS (σ, V_ρ, W_ρ) on message m under the warrant w, a verifier validates it as follows:

- checks if the message m confirms to the warrant w. Stops if not. Continues otherwise.
- checks whether the proxy signer A_0 is authorized or not in the warrant w, by the group of n original signers. Stops if not. Continues otherwise.
- obtains V_o and W_o from the warrant w, computes $h_o = H_2(w, W_o) \in Z_q^*$ and $h_\rho = H_2(m, W_\rho) \in Z_q^*$ and accepts (σ, V_ρ, W_ρ) as a valid IBPMS on message m, if and only if the following equality holds:

$$e(J, \sigma) = W_\rho V_\rho^{h_o} W_o^{h_\rho} V_o^{h_o h_\rho}.$$

4 Analysis of the Proposed Scheme

In this section, we first give the correctness of our scheme then analyze the security of our scheme and show that the proposed scheme satisfies all the security requirements of a proxy signature scheme [4].

4.1 Correctness

Correctness of the delegation verification holds since for all $0 \le i \le n$,

$$e(J, S_i) = e(J, (x_i + h_o) S_{\mathrm{ID}_{A_i}}) = e(J, S_{\mathrm{ID}_{A_i}})^{x_i + h_o} = V_i^{x_i + h_o} = W_i V_i^{h_o}. \quad (1)$$

Correctness of the IBPMS verification holds since

$$\begin{aligned} e(J, \sigma) &= e(J, (y + h_\rho) S_{pk}) = e(J, (y + h_\rho) \sum_{i=0}^{n} S_i) \\ &= \prod_{i=0}^{n} e(J, (y + h_\rho) S_i) = \prod_{i=0}^{n} e(J, S_i)^{y + h_\rho} = \prod_{i=0}^{n} \left(W_i V_i^{h_o} \right)^{y + h_\rho} \qquad \text{from (1)} \\ &= \left(\prod_{i=0}^{n} W_i V_i^{h_o} \right)^{y + h_\rho} = \left(\prod_{i=0}^{n} W_i \prod_{i=0}^{n} V_i^{h_o} \right)^{y + h_\rho} \\ &= \left(W_o V_o^{h_o} \right)^{y + h_\rho} = W_o^y (V_o^y)^{h_o} W_o^{h_\rho} (V_o^{h_o})^{h_\rho} = W_\rho V_\rho^{h_o} W_o^{h_\rho} V_o^{h_o h_\rho}. \end{aligned}$$

4.2 Security Analysis

Theorem 1. *The proposed IBPMS scheme is strongly unforgeable if the k-CAA is intractable in G_1.*

Proof. For security parameter 1^k, the challenger $\mathcal{C}$ runs the setup algorithm and provides $\langle q, G_1, P, sP, (e_1, f_1), \ldots, (e_k, f_k)\rangle$ to $\mathcal{B}$ where G_1 is an additive cyclic group of prime order q; P is a generator of G_1 and $s, e_1, \ldots, e_k \in \mathbb{Z}_q^*$ are randomly chosen elements and $f_i := \frac{1}{s+e_i}P \in G_1$, $i = 1, \ldots, k$. The goal of $\mathcal{B}$ is to solve the k-CAA problem by producing a pair $\left(e_0, \frac{1}{s+e_0}P\right)$ for some $e_0 \in \mathbb{Z}_q^*$, $e_0 \neq e_i$ for all $i = 1, \ldots, n$.

Let $\mathcal{A}$ be a forger algorithm who claims is to break the proposed identity based proxy multi-signature scheme. The adversary $\mathcal{B}$ simulates the challenger and interacts with $\mathcal{A}$. We facilitate the adversary $\mathcal{A}$ to adaptively select the identity ID^* on which it wants to forge the signature. Further the adversary can obtain the private keys associated to the identities. The adversary also can access the proxy multi-generation oracles on warrants w' of its choice, and proxy multi-signature oracles on the warrant, messages pair (w', m') of its choice upto polynomial many times.

Setup: For security parameter 1^k, $\mathcal{B}$ generates the system's public parameter $params = \langle q, G_1, G_2, e, H_1, H_2, P, Q = sP\rangle$ where G_2 is a multiplicative cyclic group of prime order q; $e : G_1 \times G_1 \to G_2$ is a bilinear map defined as in Sect. 2; and $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \times G_1 \to \mathbb{Z}_q^*$ are two cryptographic hash functions and provides *params* to $\mathcal{A}$. $\mathcal{B}$ picks a random index $i^* \in [1, k+1]$ and a random $e_0 \in \mathbb{Z}_q^*$. It then resets the values $(e_i, f_i) = (e_i, f_i)$ for $1 \leq i < i^*$, $(e_{i^*}, f_{i^*}) = (e_0, \perp)$) and $(e_i, f_i) = (e_{i-1}, f_{i-1})$ for $i^* < i \leq k+1$.
H_1-queries: To respond to the H_1 hash function queries, $\mathcal{B}$ maintains a list $L_{H_1} = \{\langle \mathrm{ID}, e, f\rangle\}$. When $\mathcal{A}$ requests the H_1 query on some identity $\mathrm{ID}_i \in \{0,1\}^*$, $i \leq k+1$, $\mathcal{B}$ responds as follows:

1. If the query ID_i already appears in the list L_{H_1} in some tuple $\langle \mathrm{ID}_j, e_j, f_j\rangle$, $j < i$, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_1(\mathrm{ID}_i) = e_j$.
   So WLOG we assume $\mathrm{ID}_i \neq \mathrm{ID}_j$ for $i \neq j$.
2. Otherwise $\mathcal{B}$ responds to $\mathcal{A}$ with $H_1(\mathrm{ID}_i) = e_i$ and adds the tuple $\langle \mathrm{ID}_i, e_i, f_i\rangle$ to the list L_{H_1}.

H_2-queries: To respond to the H_2 hash function queries, $\mathcal{B}$ maintains a list $L_{H_2} = \{\langle w, U, g\rangle\}$. When $\mathcal{A}$ requests the H_2 query on (w', U') for some $w' \in \{0,1\}^*$ and $U' \in G_1$, $\mathcal{B}$ responds as follows:

1. If the query (w', U') already appears on the list L_{H_2} in some tuple $\langle w', U', g\rangle$ then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_2(w'\|U') = g$.
2. Otherwise $\mathcal{B}$ picks a random integer $g \in \mathbb{Z}_q^*$ and adds the tuple $\langle w', U', g\rangle$ to the list L_{H_2} and responds to $\mathcal{A}$ with $H_2(w'\|U') = g$.

Extraction Queries: When $\mathcal{A}$ makes a private key query on some identity ID_i, $i \leq k+q$, $\mathcal{B}$ responds as follows:

1. If $i = i^*$, then $\mathcal{B}$ reports failure and terminates. The probability of such failure is $1/(k+1)$.

2. Otherwise $\mathcal{B}$ responds to $\mathcal{A}$ with $S_{\mathrm{ID}_i} = f_i$ and adds the tuple $\langle \mathrm{ID}_i, e_i, f_i \rangle$ to the list L_{H_1}.

Recall that, for $i \neq i^*$, $H(\mathrm{ID}_i) = e_i$ and $f_i = \frac{1}{s+e_i}P$. So, $S_{\mathrm{ID}_i} = \frac{1}{s+H(\mathrm{ID}_i)}P$ is a valid private key of the user with identity ID_i.

Delegation Queries: To respond to the delegation queries, $\mathcal{B}$ maintains a list $L_{del} = \{\langle w, (x_0, S_0), (x_1, S_1), \ldots, (x_n, S_n)\rangle\}$ and responds to identical queries in a consistent fashion. It uses L_{H_1} and L_{H_2} to generate the needed hash values and the secret keys and computes the delegations $\langle w, S_1, \ldots, S_n \rangle$ as in the actual scheme. $\mathcal{B}$ may have to terminate if the identity of one of the original signers is ID_{i^*} and the probability for that event is bounded by $(n+1)/(k+1)$.

Proxy Key Generation Queries: To respond to the proxy key generation queries, $\mathcal{B}$ maintains a list $L_{pkg} = \{\langle w, S \rangle\}$ and responds to identical queries in a consistent fashion. It uses L_{H_1} and L_{H_2} to generate the needed hash values and the secret keys and computes the proxy key $\langle w, S \rangle$ using L_{del} as in the actual scheme. $\mathcal{B}$ may have to terminate if the identity of one of the original signers or the proxy signer is ID_{i^*} and the probability for that event is $(n+1)/(k+1)$.

Proxy Multi-Signature Queries: To respond to the proxy multi-signature queries, $\mathcal{B}$ maintains a list $L_{pms} = \{\langle w, m, y, V, W, \sigma \rangle\}$ and responds to identical queries in a consistent fashion. It uses L_{H_1} and L_{H_2} to generate the needed hash values and the secret keys and computes the delegations $\langle w, m, V, W, \sigma \rangle$ using L_{del} and L_{pkg} as in the actual scheme.

$\mathcal{B}$ may have to terminate if the identity of one of the original signers or the proxy signer is ID_{i^*} and the probability for that event is $(n+1)/(k+1)$.

Output*:* $\mathcal{A}$ outputs a valid ID-based proxy multi-signature (σ, V_ρ, W_ρ) on a message m under the warrant w by the proxy signer A_0 on behalf of the group of original signers $A_1, \ldots, A_n$ such that

$$e(J, \sigma) = W_\rho V_\rho^{h_o} W_o^{h_\rho} V_o^{h_o h_\rho} \tag{2}$$

where J, V_o, W_o, h_o, h_ρ are defined as in Sect. 3.

If $\mathcal{A}$ does not query any hash function, that is, if responses to any of the hash function query is picked randomly then the probability that verification equality holds is less than $1/q$. Thus, with probability greater than $1 - 1/q$, all the public keys and were computed using H_1-oracle.

For the forgery to be valid, $\mathcal{A}$ must not have queried the private key of at least one of the signers, say A_i, and must not have received (σ, V_ρ, W_ρ) as a response to a proxy key generation query. The probability that the identity of A_i is ID_{i^*} is $1/(k+1)$ and in that case, $H_1(ID_{A_i}) = e$.

Then, using the Eq. (2) and the values returned by the adversary we can reverse compute the secret key $S_{\mathrm{ID}_{A_i}}$ of A_i as in [2]. But by definition, $S_{\mathrm{ID}_{A_i}} = \frac{1}{s+H_1(ID_{A_i})}P = \frac{1}{s+e}P$. Thus $\mathcal{B}$ can then return the pair $(e, S_{\mathrm{ID}_{A_i}})$ to the challenger $\mathcal{C}$ and win the k-CAA game.

Hence the proposed IBPMS scheme is secure.

5 Efficiency Analysis

We compare the total number of bilinear pairings (P), map-to-point hash functions (H), modular exponentiations (E) and pairing-based scalar multiplications (PSM) in Proxy key generation phase, Proxy multi-signature phase and the Verification phase with those of other IBPMS schemes [1,5,8,9] and show that our scheme is computationally more efficient and takes less operation time than the known best IBPMS schemes given in [1,5,8,9] (Table 1).

Table 1. Efficiency comparison

Proxy key generation

Scheme	P	H	E	PSM
Li et al. [5]	$3n$	n	n	$3n+1$
Wang et al. [9]	$3n$	$n+2$	0	$4n$
Shao [8]	$3n$	$n+2$	0	$2n$
Cao et al. [1]	$3n$	$n+2$	0	$2n$
Our scheme	$\mathbf{2}n$	**0**	$\mathbf{2}n$	$n\mathbf{+2}$

Proxy multi-signature

Scheme	P	H	E	PSM
Li et al. [5]	1	0	1	2
Wang et al. [9]	0	0	0	3
Shao [8]	0	1	0	2
Cao et al. [1]	0	1	0	2
Our scheme	**0**	**0**	**2**	**1**

Verification

Scheme	P	H	E	PSM
Li et al. [5]	3	$n+1$	2	0
Wang et al. [9]	3	$2n+3$	0	$n+1$
Shao [8]	4	$n+1$	0	0
Cao et al. [1]	4	$n+2$	0	0
Our scheme	**1**	**0**	**3**	**0**

Overall Time

Scheme	P	H	E	PSM
Li et al. [5]	$3n+4$	$2n+1$	$n+3$	$3n+3$
Wang et al. [9]	$3n+3$	$3n+5$	0	$5n+4$
Shao [8]	$3n+4$	$2n+4$	0	$2n+2$
Cao et al. [1]	$3n+4$	$4n+5$	0	$2n+3$
Our scheme	$\mathbf{2}n\mathbf{+1}$	**0**	$\mathbf{2}n\mathbf{+5}$	$n\mathbf{+3}$

References

1. Cao, F., Cao, Z.: A secure identity-based proxy multi-signature scheme. Inf. Sci. **179**(3), 292–302 (2009)
2. Hongzhen, D., Wen, Q.: An efficient identity-based short signature scheme from bilinear pairings. In: ICCIS 2007, pp. 725–729 (2007)
3. Gasser, M., Goldstein, A., Kaufman, C., Lampson, B.: The digital distributed system security architecture. In: NCSC 1989, pp. 305–319 (1989)
4. Lee, B., Kim, H., Kim, K.: Strong proxy signature and its applications. In: Proceedings of SCIS, vol. 1, pp. 603–608 (2001)
5. Li, X., Chen, K.: Id-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes from bilinear pairings. Appl. Math. Comput. **169**(1), 437–450 (2005)
6. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures: delegation of the power to sign messages. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **79**(9), 1338–1354 (1996)
7. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **85**(2), 481–484 (2002)

8. Shao, Z.: Improvement of identity-based proxy multi-signature scheme. J. Syst. Softw. **82**(5), 794–800 (2009)
9. Wang, Q., Cao, Z.: Identity based proxy multi-signature. J. Syst. Softw. **80**(7), 1023–1029 (2007)
10. Yi, L., Bai, G., Xiao, G.: Proxy multi-signature scheme: a new type of proxy signature scheme. Electron. Lett. **36**(6), 527–528 (2000)