Proxy Signature Without Random Oracles

Xinyi Huang, Willy Susilo, Yi Mu, and Wei Wu*

Center for Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
{xh068, wsusilo, ymu}@uow.edu.au

Abstract. In mobile Ad Hoc networks, the existence and availability of trusted authorities is severely limited by intrinsic network features, and problems such as "service availability" have become a crucial issue. A proxy signature scheme allows an entity to delegate his/her signing capability to another entity in such a way that the latter can sign messages on behalf of the former when the former is not available. This is an important primitive to ensure the service availability issue. Proxy signatures have found numerous practical applications such as distributed systems, mobile agent applications, etc. However, the security of the known proxy signature schemes is proven in the random oracle which does not imply security in the real world. In this paper, we propose the *first* proxy signature schemes *without* random oracle. The unforgeability of our scheme is based on the hardness of the well known Computational Diffie Hellman (CDH) problem.

Keywords: Proxy Signature, Without Random Oracles, CDH Problem, Bilinear Pairings.

1 Introduction

In Mobile Ad hoc Networks, permanent connections between customers and servers are unnecessary and infeasible. In order to ensure service availability to the customers distributed in the whole networks, the server must delegate his rights to some other parties in the systems, such as the mobile agents. This way, replication can be achieved and there is no need to count on a single server.

A proxy signature scheme is a variation of the standard signature schemes, in which an original signer (say, Alice) can delegate her signing right to another signer, called the proxy signer (say, Bob), for signing messages. The notion of proxy signature was introduced by Mambo, Usuda and Okamoto [15]. Since then, proxy signature schemes have attracted a considerable amount of interest from the cryptographic research community. Based on the delegation type, there are

* The work is partially supported by Nature Science Found of Jiangsu Province (No. BK2006217) and Xidian University's Open Grant of Key Laboratory on Computer Network and Information Security of Ministry of Education of China (No. 20040105).

J. Cao et al. (Eds.): MSN 2006, LNCS 4325, pp. 473–484, 2006.

three types of proxy signatures: *full delegation*, *partial delegation*, and *delegation by warrant*. In the full delegation system, Alice's secret key is given to Bob directly so that Bob can have the same signing capability as Alice. In practice, such schemes are obviously impractical and insecure. In a partial delegation proxy signature scheme, a proxy signer possesses a key, called private proxy key, which is different from Alice's private key. Hence, proxy signatures generated by using the proxy private key are different from Alice's signatures. However, in such schemes, the messages a proxy signer can sign are *not* limited. This weakness is eliminated in delegation by a warrant that specifies what kinds of messages are delegated. Here, the original signer uses the signing algorithm of a standard signature scheme and its secret key to sign a warrant and generate a signature on the warrant which is called as delegation. The proxy signer uses the delegation and his/her secret key to create a proxy signature on behalf of the original signer. According to whether the original signer can generate a valid proxy signature, proxy signatures can be classified into *proxy-unprotected* and *proxy-protected* schemes. In a proxy-protected scheme only the proxy signer can generate proxy signatures, while in a proxy-unprotected scheme either the proxy signer or the original signer can generate proxy signatures. In many applications, proxy-protected schemes are required to avoid the potential disputes between the original signer and the proxy signer. Though there exist many proxy signature schemes, most of them are insecure [14,11,13,16,17,20].

Provable security is the basic requirement for the proxy signature schemes. Currently, all the practical secure signature schemes were proven in the random oracle model. The random oracle model was introduced by Bellare and Rogaway in [5]. The model replaces hash functions by truly random objects and provides probabilistic security proofs for the resulting schemes, showing that attacks against these can be turned into efficient solutions of well-known mathematical problems, such as the discrete logarithm problem or factorization. Although the model is efficient and useful, it has received a lot of criticism that the proofs in the random oracle model are not proofs. They are simply a design validation methodology capable of spotting defective or erroneous designs when they fail. Canetti *et al.* have shown that security in the random oracle model does not imply the security in the real world in that a scheme can be secure in the random oracle model and yet be broken without violating any particular intractability assumption, and without breaking the underlying hash functions [7]. Therefore, the search for a secure proxy signature scheme without random oracle remains an open and interesting research problem.

Our Contribution

In this paper, we propose the *first* secure proxy signature scheme whose security does *not* rely on the random oracle. We incorporate Water's signature scheme [19] to obtain a concrete secure proxy signature scheme. The new scheme is proxy-protected in the sense that even the proxy signer can not forge a valid proxy signature. The security of the proposed scheme is based on the hardness of the well-known hard problem, the Computational Diffie Hellman Problem.

Roadmap
The rest of this paper is arranged as follows. In next section, we provide the preliminaries of our scheme including bilinear pairings and security assumptions. In Section 3, we describe the formal models of our proxy signature scheme. We present our proxy signature scheme without random oracle in Section 4. In Section 5, we provide formal security analysis of the proposed scheme. Finally, we conclude our paper in Section 6.

2 Preliminaries

In this section, we will review some fundamental backgrounds used throughout this paper, namely bilinear pairings and complexity assumption.

2.1 Bilinear Pairing

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two groups of prime order p and let g be a generator of $\mathbb{G}_1$. The map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear pairing if the following three conditions hold true:

- e is bilinear, i.e. $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- e is non-degenerate, i.e. $e(g, g) \neq 1_{\mathbb{G}_T}$.
- e is efficiently computable.

We say that $(\mathbb{G}_1, \mathbb{G}_T)$ are bilinear groups if there exists a group $\mathbb{G}_T$, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ as above, and e, and the group action in $\mathbb{G}_1$ and $\mathbb{G}_T$ can be computed efficiently. See [3] for more details on the construction of such pairings.

2.2 Complexity Assumption

Definition 1. Computational Diffie Hellman (CDH) Problem in $\mathbb{G}_1$
Given $g, g^a, g^b \in \mathbb{G}_1$ for some unknown $a, b \in \mathbb{Z}_p$, compute $g^{ab} \in \mathbb{G}_1$.

The success probability of a polynomial algorithm $\mathcal{A}$ in solving the CDH problem in $\mathbb{G}_1$ is denoted:

$$\mathsf{Succ}^{CDH}_{\mathcal{A},\mathbb{G}_1} = Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \in_R \mathbb{Z}_p,]$$

Definition 2. Computational Diffie Hellman (CDH) Assumption in $\mathbb{G}_1$
Given $g, g^a, g^b \in_R \mathbb{G}_1$, for some unknown $a, b \in \mathbb{Z}_p$, $\mathsf{Succ}^{CDH}_{\mathcal{A},\mathbb{G}_1}$ is negligible.

3 Formal Models of Proxy Signatures

Let Alice denote the original signer and Bob the proxy signer. Our proxy signature scheme consists of the following algorithms: ParaGen, KeyGen, StandardSign, DelegationGen, ProxySign and ProxyVerification.

1. ParaGen: Taking as input the system security parameter ℓ, this algorithm outputs system's parameters: Para. That is: Para $\leftarrow$ ParaGen(ℓ)
2. KeyGen: Taking as input system's parameter Para, this algorithm generates a secret-public key pair (sk_i, pk_i) where $i \in \{a, b\}$ denotes Alice and Bob, respectively. That is: $(sk_i, pk_i) \leftarrow$ KeyGen(Para)
3. StandardSign: Input system's parameter Para, the signer's secret key sk and the message M to be singed, this algorithm generates the standard signature: σ_S. That is: $\sigma_S \leftarrow$ StandardSign (Para, sk, M)
4. DelegationGen: Input system's parameter Para, the original signer's secret key sk_a and the warrant W to be singed, this algorithm uses the StandardSign algorithm to generate the delegation: σ_W. That is: $\sigma_W \leftarrow$ DelegationGen(Para, sk_a, W)
5. ProxySign: Input system's parameter Para, the warrant W, the delegation σ_w, the secret key sk_b of the proxy signer and the message M to be signed, this algorithm generates the proxy signature σ. That is: $\sigma_M \leftarrow$ ProxySign(Para, W, σ_W, sk_b, M)
6. ProxyVerification: Input system's parameter Para, original signer's public keys pk_a, proxy signer's public key pk_b, the warrant W, the signed message M and the signature σ_M, this algorithm outputs True if σ is a valid proxy signature of the message M and output $\perp$ otherwise. That is: $\{\text{True}, \perp\} \leftarrow$ ProxyVerification(Para, $pk_a, pk_b, W, M, \sigma_M$)

3.1 Security Models

Lee, Kim and Kim defined some properties that a strong proxy signature scheme should provide in [12]. While these informal requirements provide some intuition about the goals that a notion of security for proxy signature schemes should capture, their precise meaning is unclear. The first security model of proxy signature was proposed in [4]. In [10], the authors also proposed a security model of the proxy signature. In the model defined in[10], they divide the potential attackers into three kinds:

1. **Type I:** This type adversary $\mathcal{A}_I$ only has the public keys of Alice and Bob.
2. **Type II:** This type adversary $\mathcal{A}_{II}$ has the public keys of Alice and Bob, he additionally has the secret key of the proxy signer Bob.
3. **Type III:** This type adversary $\mathcal{A}_{III}$ has the public keys of Alice and Bob, he additionally has the secret key of the original signer Alice.

One can find that if a proxy signature scheme is secure against Type II (or Type III) adversary, the scheme is also secure against Type I adversary. We note the above classification helps to make the security model clearer, therefore, we will use this classification to redefine and improve the security model proposed in [4]. In the security model defined later, we only consider the general case of the proxy signature where the original signer and the proxy signer are distinct.

In a warrant based proxy signature, the delegation is the original signer's standard signature on the warrant which contains information regarding the particular proxy signer such as the proxy signer's public key, a period of validity,

and restrictions on the class of messages for which the warrant is valid. Therefore, this kind of proxy signature can prevent the misuse of the delegation. Here after, we only focus on the unforgeability of the proxy signature.

Existential unforgeability against adaptive A_{II} Adversary
Roughly speaking, the existential unforgeability of a proxy signature scheme under a type II attacker requires that it is difficult for a user to forge a valid proxy signature under a warrant if he does not obtain the delegation of this warrant. It is defined using the following game between the challenger $\mathcal{C}$ and a type II adversary $\mathcal{A}_{II}$:

- Setup: $\mathcal{C}$ runs the ParaGen algorithm to obtain system's parameter Para, runs KeyGen to obtain the secret-public key pairs $(sk_a, pk_a), (sk_b, pk_b)$ of the original signer Alice and proxy signer Bob, respectively. $\mathcal{C}$ then sends (pk_a, pk_b, sk_b) to the adversary A_{II}.
- Delegation queries: Proceeding adaptively, A_{II} can request the delegation on the warrant W. In response, $\mathcal{C}$ runs the DelegationGen algorithm to obtain σ_W and returns σ_W to the adversary A_{II}.
- ProxySign queries: Proceeding adaptively, A_{II} can request the proxy signature on the message M under the warrant W. In response, $\mathcal{C}$ runs DelegationGen algorithm to generate the delegation on the warrant W. Then $\mathcal{C}$ runs the ProxySign algorithm to obtain the proxy signature σ_M and returns σ_M to the adversary A_{II}.
- Output: Finally, A_{II} outputs a signature σ^* with the warrant W^* and the message M^* such that
  1. W^* has not been requested as one of the Delegation queries.
  2. (M^*, W^*) has not been requested as one of the ProxySign queries.
  3. σ^* is a valid proxy signature of the message M^* under the warrant W^*.

Compared with the model defined in [4], an important refinement is that $\mathcal{A}_{II}$ can adaptively submit the ProxySign queries under warrant whose delegation is unknown to $\mathcal{A}_{II}$. The only restrictions are that when $\mathcal{A}_{II}$ outputs the forgery (M^*, W^*, σ^*), he cannot submit W^* as one of the Delegation queries or submit (M^*, W^*) as one of the ProxySign queries. However, he can even submit (M', W^*) to the ProxySign queries where $M' \neq M^*$. The success probability of an algorithm A_{II} wins the above game is defined as $Succ\ A_{II}$.

Definition 3. *We say a type II adversary $\mathcal{A}_{II}$ can $(t, q_W, q_{PS}, \varepsilon)$ break a proxy signature scheme if A_{II} runs in time at most t, $\mathcal{A}_{II}$ makes at most q_W* Delegation *queries and at most q_{PS}* ProxySign *queries and Succ A_{II} is at least ε.*

Existential unforgeability against adaptive A_{III} adversary
The existential unforgeability of a proxy signature scheme under a type III attacker requires that it is difficult for the original signer to generate a valid proxy signature of a message M^* which has not been singed by the proxy signer. It is defined using the following game between the challenger $\mathcal{C}$ and a type III adversary $\mathcal{A}_{III}$:

- Setup: $\mathcal{C}$ runs the ParaGen algorithm to obtain system's parameter Para, runs KeyGen to obtain the secret-public key pairs $(sk_a, pk_a), (sk_b, pk_b)$ of the original signer Alice and proxy signer Bob, respectively. $\mathcal{C}$ then sends (pk_a, pk_b, sk_a) to the adversary $\mathcal{A}_{III}$.
- StandardSign: Proceeding adaptively, $\mathcal{A}_{III}$ can request proxy signer's standard signature on the message M. In response, $\mathcal{C}$ runs the StandardSign algorithm to generate the standard signature on the message M and returns to the adversary $\mathcal{A}_{III}$.
- ProxySign queries: Proceeding adaptively, $\mathcal{A}_{III}$ can request the proxy signature on the message M under the warrant W. In response, $\mathcal{C}$ runs the DelegationGen algorithm to generate the delegation on the warrant W. Then $\mathcal{C}$ runs the ProxySign algorithm to generate the proxy signature σ_M and returns σ_M to the adversary $\mathcal{A}_{III}$.
- Output: Finally, $\mathcal{A}_{III}$ outputs a signature σ^* with the warrant W^* and the message M^* such that
  1. (M^*, W^*) has not been requested as one of the ProxySign queries.
  2. σ^* is a valid proxy signature of the message M^* under the warrant W^*.

In this model, we allow the attacker $\mathcal{A}_{III}$ can submit StandardSign queries, this is to guarantee that proxy signer's standard signature on the message M^* can not help the attacker to forge a valid proxy signature on the same message. The success probability of an algorithm $\mathcal{A}_{III}$ wins the above game is defined as $Succ\ \mathcal{A}_{III}$

Definition 4. *We say a type III adversary $\mathcal{A}_{III}$ can $(t, q_S, q_{PS}, \varepsilon)$ break a proxy signature scheme if A_{III} runs in time at most t, $\mathcal{A}_{III}$ makes at most q_S* StandardSign *queries and q_{PS}* ProxySign *queries, and Succ A_{III} is at least ε.*

4 Proposed Scheme

In this section, we will describe our proxy signature scheme without random oracle. It consists of the following algorithms:

1. ParaGen: Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups defined in Section 2.1 where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$ for some prime p, g is the generator of $\mathbb{G}_1$. e denotes the bilinear pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The messages M to be signed in this scheme will be represented as bitstrings of length n. Furthermore, picks $2n + 2$ random elements $u', v', u_1, u_2, \cdots, u_n, v_1, \cdots, v_n \in_R \mathbb{G}_1$ and set $\boldsymbol{u} = (u_1, u_2, \cdots, u_n)$, $\boldsymbol{v} = (v_1, v_2, \cdots, v_n)$. Then the common parameter Para $= (\mathbb{G}_1, \mathbb{G}_T, p, g, e, n, u', v', \boldsymbol{u}, \boldsymbol{v})$.
2. Key Gen: The original Alice picks two secret values $x_a, y_a \in_R \mathbb{Z}_p^*$ and set the secret key $sk_a = (sk_{ax}, sk_{ay}) = (x_a, y_a)$. Then the signer computes the public key $pk_a = (pk_{ax}, pk_{ay}) = (g^{x_a}, g^{y_a})$. Similarly, the proxy signer's secret key is $sk_b = (sk_{bx}, sk_{by}) = (x_b, y_b)$ and the public key is $pk_b = (pk_{bx}, pk_{by}) = (g^{x_b}, g^{y_b})$

3. StandardSign: Let M be an n-bit message to be signed and M_i denote the i^{th} bit of M, and $\mathcal{M} \in \{1, \cdots, n\}$ be the set of all i for which $M_i = 1$, the standard signature is generated as follows. First, a random $r \in \mathbb{Z}_p$ is chosen. Then the standard signature is constructed as: $\sigma_S = (\sigma_{S_1}, \sigma_{S_2})$ where $\sigma_{S_1} = g^{sk_x sk_y}(u' \prod_{i \in \mathcal{M}} u_i)^r, \sigma_{S_2} = g^r$. Here sk_x, sk_y denote the secret key of the signer.
4. DelegationGen: Let W be an n-bit message to be signed by the original signer Alice and W_i denote the i^{th} bit of W, and $\mathcal{W} \in \{1, \cdots, n\}$ be the set of all i for which $W_i = 1$, the delegation is generated as follows. First, a random $r_a \in \mathbb{Z}_p$ is chosen. Then the signature is constructed as: $\sigma_W = (\sigma_{W_1}, \sigma_{W_2})$ where $\sigma_{W_1} = g^{x_a y_a}(u' \prod_{i \in \mathcal{W}} u_i)^{r_a}, \sigma_{W_2} = g^{r_a}$. Then Alice sends the delegation σ_W with the warrant W to the proxy signer Bob.
5. ProxySign: Let M be an n-bit message to be signed by the original signer Alice and M_i denote the i^{th} bit of M, and $\mathcal{M} \in \{1, \cdots, n\}$ be the set of all i for which $M_i = 1$, the proxy signature is generated as follows. First, two random values $r'_a, r_b \in \mathbb{Z}_p$ are chosen. Then the signature is constructed as:

$$\sigma_M = (\sigma_{M_1}, \sigma_{M_2}, \sigma_{M_3}) = (\sigma_{W_1}(u' \prod_{i \in \mathcal{W}} u_i)^{r'_a} g^{x_b y_b}(v' \prod_{i \in \mathcal{M}} v_i)^{r_b}, \sigma_{W_2} g^{r'_a}, g^{r_b}).$$
$$= (g^{x_a y_a} g^{x_b y_b}(u' \prod_{i \in \mathcal{W}} u_i)^{r_a + r'_a}(v' \prod_{i \in \mathcal{M}} v_i)^{r_b}, g^{r_a + r'_a}, g^{r_b})$$

6. Verification: Given the public keys (pk_a, pk_b), a warrant $W \in \{0,1\}^n$, a message $M \in \{0,1\}^n$, and a signature $\sigma_M = (\sigma_{M_1}, \sigma_{M_2}, \sigma_{M_3})$, verify whether

$$e(\sigma_{M_1}, g) \stackrel{?}{=} e(pk_{ax}, pk_{ay}) e(pk_{bx}, pk_{by}) e(u' \prod_{i \in \mathcal{W}} u_i, \sigma_{M_2}) e(v' \prod_{i \in \mathcal{M}} v_i, \sigma_{M_3}).$$

If the equality holds the result is True; otherwise the result is $\perp$.

Correctness:

$$\begin{aligned} e(\sigma_{M_1}, g) &= e(g^{x_a y_a} g^{x_b y_b}(u' \prod_{i \in \mathcal{W}} u_i)^{r_a + r'_a}(v' \prod_{i \in \mathcal{M}} v_i)^{r_b}, g) \\ &= e(g^{x_a y_a}, g) e(g^{x_b y_b}, g) e((u' \prod_{i \in \mathcal{W}} u_i)^{r_a + r'_a}, g) e((v' \prod_{i \in \mathcal{M}} v_i)^{r_b}, g) \\ &= e(pk_{ax}, pk_{ay}) e(pk_{bx}, pk_{by}) e(u' \prod_{i \in \mathcal{W}} u_i, g^{r_a + r'_a}) e(v' \prod_{i \in \mathcal{M}} v_i, g^{r_b}) \\ &= e(pk_{ax}, pk_{ay}) e(pk_{bx}, pk_{by}) e(u' \prod_{i \in \mathcal{W}} u_i, \sigma_{M_2}) e(v' \prod_{i \in \mathcal{M}} v_i, \sigma_{M_3}) \end{aligned}$$

5 Security Analysis

In this section, we will provide the formal security analysis of the proposed proxy signature scheme.

5.1 Unforgeability Against Type II Adversary

Theorem 1. *If there exists a type II adversary $\mathcal{A}_{II}$ can $(t, q_W, q_{PS}, \varepsilon)$ breaks the proposed proxy signature scheme then there exists another algorithm $\mathcal{B}$ who can use $\mathcal{A}_{II}$ to solve an instance of the CDH problem in $\mathbb{G}_1$ with the probability*

$$Succ^{CDH}_{\mathcal{B},\mathbb{G}_1} \geq \frac{\varepsilon}{27(n+1)^2} \cdot \frac{1}{(q_W + q_{PS})^2}$$

in time $t + c_1(4q_W + 7q_{PS}) + c_2((n+2)q_W + (2n+4)q_{PS})$. Here c_1, c_2 are the two constants that depend on $\mathbb{G}_1$.

Proof. Let $\mathbb{G}_1$ be a bilinear pairing group of prime order p. Algorithm $\mathcal{B}$ is given $g, g^a, g^b \in \mathbb{G}_1$ which is a random instance of the CDH problem. Its goal is to compute g^{ab}. Algorithm $\mathcal{B}$ will simulate the challenger and interact with the forger $\mathcal{A}_{II}$ as described below.

Let's recall the definition of the type II adversary $\mathcal{A}_{II}$. This type of adversary $\mathcal{A}_{II}$ has the public key of the original signer Alice and the proxy singer Bob, he also has Bob's secret key.

1. Setup: $\mathcal{B}$ chooses two integers ℓ_a, ℓ_b, and other two integers, k_a, k_b, uniformly at random between 0 and n. Then it chooses two values x'_a, x'_b and two random n-vectors, $\boldsymbol{x_a} = (x_{ai}), \boldsymbol{x_b} = (x_{bi})$ where $x'_a, x_{ai} \in_R \mathbb{Z}_{\ell_a}, x'_b, x_{bi} \in_R \mathbb{Z}_{\ell_b}$. Additionally, $\mathcal{B}$ chooses two values y'_a, y'_b and two random n-vectors $\boldsymbol{y_a} = (y_{ai}), \boldsymbol{y_b} = (y_{bi})$ where $y'_a, y'_b, y_{ai}, y_{bi} \in_R \mathbb{Z}_p$. $\mathcal{B}$ keeps all the values secret.

   For an n-bit X, we let $\mathcal{X} \subseteq \{1, 2, \cdots, n\}$ be the set of all i for which $X_i = 1$. Then, for a warrant W, $\mathcal{W}$ be the set of all i for which $W_i = 1$. Similarly, for a message M, $\mathcal{M}$ be the set of all i for which $M_i = 1$. To make the notation easy to follow, we define six functions $F_a(X), F_b(X), J_a(X), J_b(X)$ and $K_a(X), K_b(X)$ as [19]:

   (a) $F_a(X) = (p - \ell_a k_a) + x'_a + \Sigma_{i \in \mathcal{X}} x_{ai}$ and $F_b(X) = (p - \ell_b k_b) + x'_b + \Sigma_{i \in \mathcal{X}} x_{bi}$

   (b) $J_a(X) = y'_a + \Sigma_{i \in \mathcal{X}} y_{ai}$ and $J_b(M) = y'_b + \Sigma_{i \in \mathcal{X}} y_{bi}$

   (c) $K_a(X) = \begin{cases} 0, & \text{if } x'_a + \Sigma_{i \in \mathcal{X}} x_{ai} \equiv 0 \pmod{\ell_a} \\ 1, & \text{otherwise} \end{cases}$

   and $K_b(X) = \begin{cases} 0, & \text{if } x'_b + \Sigma_{i \in \mathcal{X}} x_{bi} \equiv 0 \pmod{\ell_b} \\ 1, & \text{otherwise} \end{cases}$

   $\mathcal{B}$ sets the public keys of the users and the common parameter as:

   (a) $\mathcal{B}$ chooses two random numbers $sk_{bx}, sk_{by} \in \mathbb{Z}_p^*$ and sets

   $$pk_{ax} = g^a, pk_{ay} = g^b, pk_{bx} = g^{sk_{bx}}, pk_{by} = g^{sk_{by}}.$$

   Where g^a, g^b are the input of the CDH problem.

   (b) $\mathcal{B}$ assigns $u' = pk_{ay}^{p - k_a \ell_a + x'_a} g^{y'_a}, u_i = pk_{ay}^{x_{ai}} g^{y_{ai}}, \boldsymbol{u}_a = (u_1, u_2, \cdots, u_n)$

   (c) $\mathcal{B}$ then assigns, $v' = pk_{ay}^{p - k_b \ell_b + x'_b} g^{y'_b}, v_i = pk_{by}^{x_{bi}} g^{y_{bi}}$ and $\boldsymbol{v} = (v_1, v_2, \cdots, v_n)$. Then $\mathcal{B}$ returns $(\mathbb{G}_1, \mathbb{G}_T, e, p, g, \boldsymbol{u}, u', \boldsymbol{v}, v')$ and $(pk_{ax}, pk_{ay}, pk_{bx}, pk_{by}, sk_{bx}, sk_{by})$ to the Type II adversary $\mathcal{A}_{II}$.

2. Delegation queries: Suppose $\mathcal{A}_{II}$ issues a delegation query for an n-bit warrant W. If $K_a(W) \neq 0$ (If we have $K_a(W) \neq 0$ this implies $F_a(W) \neq 0 \pmod p$, since we can assume $p > n\ell_a$ for any reasonable values of p, n, and ℓ_a[19]), $\mathcal{B}$ can construct the delegation of this warrant by choosing a random $r_a \in \mathbb{Z}_p$ and computing:

$$\sigma_W = (\sigma_{W_1}, \sigma_{W_2}) = \left(pk_{ax}^{\frac{-J_a(W)}{F_a(W)}} (u' \prod_{i \in \mathcal{W}} u_i)^{r_a}, pk_{ax}^{\frac{-1}{F_a(W)}} g^{r_a} \right)$$

If $K_a(W) = 0$. $\mathcal{B}$ terminates the simulation and reports failure.

3. ProxySign queries: Suppose $\mathcal{A}_{II}$ issues a delegation query for an n-bit message M under the warrant W.
   (a) If $K_a(W) = 0, K_b(M) = 0$, $\mathcal{B}$ terminates the simulation and reports failure.
   (b) Else $K_a(W) = 0, K_b(M) \neq 0$, $\mathcal{B}$ can construct the delegation of this warrant by choosing a random $r_a, r_b \in \mathbb{Z}_p$ and computing: $\sigma_M = (\sigma_{M_1}, \sigma_{M_2}, \sigma_{M_3})$. where

$$\sigma_{M_1} = \left(pk_{ax}^{\frac{-J_b(M)}{F_b(M)}} (u' \prod_{i \in \mathcal{W}} u_i)^{r_a} \cdot g^{sk_{bx} sk_{by}} (v' \prod_{i \in \mathcal{M}} v_i)^{r_b} \right),$$

$$\sigma_{M_2} = g^{r_a}, \sigma_{M_3} = pk_{ax}^{\frac{-1}{F_b(M)}} \cdot g^{r_b}$$

   (c) Otherwise $K_a(W) \neq 0$. In this case, $\mathcal{B}$ can compute the delegation of the warrant W as he does in response to the delegation queries. Since $\mathcal{B}$ knows the secret key sk_{bx}, sk_{by} of proxy signer, $\mathcal{B}$ can run the ProxySign algorithm as defined in Section 4 to compute the proxy signature and return the signature to $\mathcal{A}_{III}$.

Finally, the adversary $\mathcal{A}_{II}$ outputs a proxy signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$ of the message M^* under the warrant W^* such that

(a) W^* has not been submitted as one of the Delegation queries.
(b) (M^*, W^*) has not been submitted as one of the ProxySign queries.
(c) $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$ is a valid signature, that is:

$$\sigma_1^* = g^{sk_{ax} sk_{ay}} g^{sk_{bx} sk_{by}} (u' \prod_{i \in \mathcal{W}^*} u_i)^{r_a^*} (v' \prod_{i \in \mathcal{M}^*} v_i)^{r_b^*}, \sigma_2^* = g^{r_a^*}, \sigma_3^* = g^{r_b^*}$$

If $F_a(W^*) \neq 0$ or $F_b(M^*) \neq 0$, $\mathcal{B}$ will abort. Otherwise, $F_a(W^*) = 0$, $F_b(M^*) = 0$. In this case,

$$\begin{aligned}
\sigma_1^* &= g^{sk_{ax} sk_{ay}} g^{sk_{bx} sk_{by}} (u' \prod_{i \in \mathcal{W}^*} u_i)^{r_a^*} (v' \prod_{i \in \mathcal{M}^*} v_i)^{r_b^*} \\
&= g^{ab} g^{sk_{bx} sk_{by}} (g^{J_a(W^*)})^{r_a^*} (g^{J_b(M^*)})^{r_b^*} \\
&= g^{ab} g^{sk_{bx} sk_{by}} (g^{r_a^*})^{J_a(W^*)} (g^{r_b^*})^{J_b(M^*)} \\
&= g^{ab} g^{sk_{bx} sk_{by}} (\sigma_2^*)^{J_a(W^*)} (\sigma_3^*)^{J_b(M^*)}
\end{aligned}$$

Therefore, $\mathcal{B}$ can compute $g^{ab} = \frac{\sigma_1^*}{g^{sk_{bx} sk_{by}} (\sigma_2^*)^{J_a(W^*)} (\sigma_3^*)^{J_b(M^*)}}$

This completes the description of the simulation. It remains to analyze the probability of $\mathcal{B}$ not aborting. $\mathcal{B}$ will not abort if all the following cases happen:

$\mathsf{A}: K_a(W) \neq 0 \pmod{\ell_a}$ during Delegation queries
$\mathsf{B}: K_a(W) \neq 0 \pmod{\ell_a}$ or $K_b(M) \neq 0 \pmod{\ell_b}$ during ProxySign queries
$\mathsf{C}: F_a(W^*) = 0 \pmod{p}$ and $F_b(M^*) = 0 \pmod{p}$

The success probability is $Succ_{\mathcal{B}}^{CDH}$=Pr$[A \wedge B \wedge C]\varepsilon$.

$$\begin{aligned}
&\Pr[A \wedge B \wedge C] = \Pr[\bigwedge_{i=1}^{q_W} K_a(W_i) \neq 0 \bigwedge_{i=1}^{q_{PS}} \Big(K_a(W_i) \neq 0 \bigvee K_b(M_i) \neq 0\Big) \\
&\bigwedge F_a(W^*) = 0 \pmod{p} \bigwedge F_b(M^*) = 0 \pmod{p}] \\
&\geq \frac{1}{(n+1)^2 \ell_a \ell_b}(1 - \frac{2(q_W + q_{PS})}{\ell_a})
\end{aligned}$$

Therefore, $Succ_{\mathcal{B},\mathbb{G}_1}^{CDH} \geq \frac{1}{(n+1)^2 \ell_a \ell_b}(1 - \frac{2(q_W+q_{PS})}{\ell_a})\varepsilon$. We can optimize it by setting $\ell_a = \ell_b = 3(q_W + q_{PS})$, then

$$Succ_{\mathcal{B},\mathbb{G}_1}^{CDH} \geq \frac{\varepsilon}{27(n+1)^2} \cdot \frac{1}{(q_W + q_{PS})^2}$$

Algorithm $\mathcal{B}$'s running time is the same as $\mathcal{A}'_{II}s$ running time plus the time it takes to respond to q_W Delegation queries and q_{PS} ProxySign queries. Each Delegation query requires 4 exponentiation operations and $n+2$ multiplication operations in $\mathbb{G}_1$. Each ProxySign query requires at most 7 exponentiation operations and $2n+4$ multiplication operations in $\mathbb{G}_1$. If we assume each exponentiation takes time c_1 and each multiplication takes time c_2, the total running time is at most $t + c_1(4q_W + 7q_{PS}) + c_2((n+2)q_W + (2n+4)q_{PS})$. This completes the proof. □

5.2 Unforgeability Against Type III Adversary

Theorem 2. *If there exists a type III adversary $\mathcal{A}_{III}$ can $(t, q_S, q_{PS}, \varepsilon)$ breaks the proposed proxy signature scheme then there exists another algorithm $\mathcal{B}$ who can use $\mathcal{A}_{III}$ to solve an instance of the CDH problem in $\mathbb{G}_1$ with the probability*

$$Succ_{\mathcal{B},\mathbb{G}_1}^{CDH} \geq \frac{\varepsilon}{27(n+1)^2} \cdot \frac{1}{(q_S + q_{PS})^2}$$

in time $t + c_1(4q_W + 7q_{PS}) + c_2((n+2)q_W + (2n+4)q_{PS})$. Here c_1, c_2 are the two constants that depend on $\mathbb{G}_1$.

Proof. It is similar to the proof of Theorem 1.

6 Conclusion

In this paper, we proposed the first proxy signature scheme without random oracle based on Water's signature scheme [19]. We showed that our scheme is unforgeable against an adaptively chosen message attacker. Even the original signer can not forge a valid proxy signature of our scheme. The security of our scheme is based on the Computational Diffie Hellman problem.

Acknowledgement

The authors would like to thank the anonymous referees of the Second International Conference on Mobile Ad Hoc and Sensor Networks (MSN 2006) for the suggestions to improve this paper.

References

1. D. Boneh and X. Boyen. Short signatures without random oracles. In Advances in Cryptology, Proc. EUROCRYPT 2004, Lecture Notes in Computer Science 3027, pages. 56–73. Springer–Verlag, 2004.
2. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In Advances in Cryptology, Proc. CRYPTO 2001, Lecture Notes in Computer Science 2139, pages. 213–229. Springer–Verlag, 2001.
3. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In Advances in Cryptology–ASIACRYPT 2001, Lecture Notes in Computer Science 2248, pages. 514–532. Springer–Verlag, 2001.
4. A. Boldyreva, A. Palacio and B. Warinschi. Secure proxy signature scheme for delegation of signing rights. IACR ePrint Archive, available at `http://eprint.iacr.org/2003/096/`, 2003.
5. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. Advances in Cryptology - Eurocrypt'96, Lecture Notes in Computer Science 950, pages 399–416, Springer-Verlag, Berlin, 1996.
6. J. H. Cheon. Security analysis of the strong diffie-hellman problem. EUROCRYPT 2006, to appear.
7. R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, revisited. In Proceedings of the 30th Annual Symposium on the Theory of Computing (STOC'98), pages 209–218, 1998.
8. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptively chosen message attacks. SIAM Journal on Computing, 17(2):281-308, 1988
9. S.D. Galbraith and K.G. Paterson and N.P. Smart. Pairings for cryptographers. IACR ePrint Archive, available at http://eprint.iacr.org/2006/165, 2006.
10. X. Huang, Y. Mu, W. Susilo, F. Zhang and X. Chen. A short proxy signature scheme: efficient authentication in the ubiquitous world. The Second International Symposium on Ubiquitous Intelligence and Smart Worlds (UISW2005), Lecture Notes in Computer Science 3823, pages. 480–489, Springer-Verlag, 2005.
11. J.-Y. Lee, J. H. Cheon and S. Kim. An analysis of proxy signatures: Is a secure channel necessary? In Topics in Cryptology - CT-RSA 2003, Lecture Notes in Computer Science 2612, pages. 68–79. Springer–Verlag, 2003.

12. B. Lee, H. Kim and K. Kim. Strong proxy signature and its applications. In Proc of SCIS'01, pages. 603–08. 2001.
13. B. Lee, H. Kim, and K. Kim. Secure mobile agent using strong nondesignated proxy signature. In Information Security and Privacy (ACISP01), Lecture Notes in Computer Science 2119, pages. 474–486. Springer–Verlag, 2001.
14. S. Kim, S. Park and D. Won. Proxy signatures, revisited. In Information and Communications Security (ICICS97), Lecture Notes in Computer Science 1334, pages. 223–232. Springer–Verlag, 1997.
15. M. Mambo, K. Usuda and E. Okamoto. Proxy signature: delegation of the power to sign messages. IEICE Trans. Fundamentals, Vol. E79-A, No. 9, Sep., pages. 1338–1353, 1996.
16. T. Okamoto, A. Inomata, and E. Okamoto. A proposal of short proxy signature using pairing. In International Conference on Information Technology (ITCC 2005), pages. 631–635. IEEE Computer Society, 2005.
17. T. Okamoto, M. Tada, and E. Okamoto. Extended proxy signatures for smart cards. In ISW 99, Lecture Notes in Computer Science 1729, pages. 247–258, Springer-Verlag, 1999.
18. H.-U. Park and I.-Y. Lee. A digital nominative proxy signature scheme for mobile communications. In Information and Communications Security (ICICS 2001), Lecture Notes in Computer Science 2229, pages. 451–455, Springer–Verlag, 2001.
19. R. Waters. Efficient identity-based encryption without random oracles. In EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pages. 114–127. Springer–Verlag, 2005.
20. G. Wang, F. Bao, J. Zhou and Robert H. Deng. Security analysis of some proxy signatures. In ICICS 2003, Lecture Notes in Computer Science 2971, pages. 305–319. Springer–Verlag, 2003.
21. R. Zhang, J. Furukawa and H. Imai. Short signature and universal designated verifier signature without random oracles. In Applied Cryptography and Network Security (ACNS 2005), Lecture Notes in Computer Science 3531, pages. 483–498. Springer-Verlag, 2005.
22. F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In Public Key Cryptography (PKC'04), Lecture Notes in Computer Science 2947, pages 277–290. Springer–Verlag, 2004.