A Short Proxy Signature Scheme: Efficient Authentication in the Ubiquitous World*

Xinyi Huang[1], Yi Mu[2], Willy Susilo[2], Fangguo Zhang[3], and Xiaofeng Chen[4]

[1] College of Mathematics and Computer Science,
Nanjing Normal University, P.R. China
xinyinjnu@126.com
[2] Centre for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong, Australia
{wsusilo, ymu}@uow.edu.au
[3] Department of Electronics and Communication Engineering,
Sun Yat-Sen University, Guangzhou 510275, P.R. China
isdzhfg@zsu.edu.cn
[4] Department of Computer Science,
Sun Yat-Sen University, Guangzhou 510275, P.R. China
isschxf@zsu.edu.cn

Abstract. We present a cryptanalysis on the short proxy signature scheme recently proposed in [11] and propose a novel short proxy signature scheme from bilinear pairings. Compared with the existing proxy signature schemes, the signature length of our scheme is the *shortest*. Our short proxy signature scheme satisfies all the properties required for proxy signatures. We prove that our scheme is secure in the random oracle model.

Keywords: Proxy Signature, Short Signature, Authentication.

1 Introduction

Ubiquitous computing plays an important role in many aspects such as human factors, computer science, engineering, and social sciences. However, Placing computers in human life would face an essential problem, namely, how to implement security and trust among the users that connected to a network. As an example, in a wireless network, the users can connect to a network in anywhere within the broadcast power range. How can they know that they are talking with a real person? Therefore, a necessary authentication scheme must be deployed. In a distributed computing environment, usually, a network is heavily loaded with thousands of users and the bandwidth consumption is a major concern. To

* This work is supported by ARC Discovery Grant DP0557493 and the National Natural Science Foundation of China 60403007.

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 480–489, 2005.

achieve security without consuming substantial bandwidth is a major challenge to security researchers. In this paper, we will describe an authentication scheme that presents a promise to the minimal use of bandwidth and to providing strong authentication in a "proxy" environment.

The concept of proxy signature can be very useful in cases when a user (say, Alice) wants to delegate her signing right to the other user or proxy (say, Bob). Once the delegation is performed, the proxy can then sign on behalf of the original signer. The notion of proxy signature was introduced by Mambo, Usuda and Okamoto [10]. Based on the delegation type, they classified proxy signatures as full delegation, partial delegation, and delegation by warrant. In the full delegation system, Alice's private 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 proxy signature scheme, a proxy signer possesses a key, called private proxy key, which is different from Alice's private key. So, proxy signatures generated by using the proxy key are different from Alice's signatures. However, in such schemes, the messages a proxy signer can sign is not limited. This weakness is eliminated in delegation by a warrant that specifies what kinds of messages are delegated. Some related works about proxy signatures can be found from [4, 9, 8, 6, 12].

According to whether the original signer knows the proxy private key, proxy signatures can be classified into proxy-unprotected and proxy-protected. 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 since both of them has a knowledge on the proxy private key. In many applications, proxy-protected schemes are required to avoid the potential disputes between the original signer and the proxy signer.

Short signature has attracted a lot of attention since the exploring positive use of bilinear pairing [2]. With bilinear pairings, a digital signature can be as short as 160 bits [3,1,5]. Short signatures have a great advantage while the bandwidth of a communication channel is limited. Recently, Okamoto, Inomata and Okamoto [11] proposed a short proxy signature scheme, which allows a much shorter size than other existing schemes. We refer it to as OIO scheme. Unfortunately, we found that the scheme is flawed.

In this paper, we show that their scheme is not secure against a dishonest original signer; namely, given a valid proxy signature, the original signer can forge a valid proxy signature of any new message. We then propose a novel proxy signature, which, we believe, is the shortest proxy signature scheme amongst all existing proxy signature schemes. We also provide a security proof to our novel scheme and show that our scheme is secure against dishonest Alice and Bob and any other polynomial-time adversaries.

The rest of this paper is arranged as follows. In Section 2, we provide the preliminaries of our scheme including bilinear pairings and security assumptions. In Section 3, we give a cryptanalysis on the OIO scheme and show that their scheme is flawed. In Section 4, we describe the model of our proxy signature scheme. In Section 5, we present a novel construction of the shortest proxy

signature. In Section 6, we provide a security proof to our scheme. We show that our scheme is secure against any polynomial adversaries. In the last section, we conclude our paper.

2 Preliminaries

2.1 Basic Concepts on Bilinear Pairings

Let $\mathbb{G}_1$ be cyclic additive groups of prime order q and is generated by P. Let $\mathbb{G}_2$ be a cyclic multiplicative group with the same order q. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_2$ be a bilinear mapping with the following properties:

1. *Bilinearity*: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_1$ and $a, b, \in \mathbb{Z}_q$. Here $\mathbb{Z}_q$ denotes the definite field of the order q.
2. *Non-Degeneracy*: There exists $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_2}$.
3. *Computability*: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_1$.

2.2 Security Assumption

Definition 1. Computational Diffie-Hellman (CDH) Problem.
Given two randomly chosen $aP, bP \in (\mathbb{G}_1, +)$ of prime order q, for unknown $a, b \in \mathbb{Z}_q$, compute $Z = abP$.

The CDH assumption states that for every probabilistic polynomial-time algorithm $\mathcal{A}$, $\mathsf{Succ}^{CDH}_{\mathcal{A},\mathbb{G}_1}$ is negligible.

2.3 ZSS Signature Scheme [5]

The ZSS signature scheme proposed in [5] consists of the following algorithms: a parameter generation algorithm ParamGen, a key generation algorithm KeyGen, a signature generation algorithm Sign and a signature verification algorithm Ver.

1. ParamGen: The system parameters are $\{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, H\}$. Here $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ is a cryptographic hash function
2. KeyGen: Randomly selects $x \in \mathbb{Z}_q^*$, and computes $P_{pub} = xP$. The public key of the signer is P_{pub}. The secret key is x.
3. Sign: Given a secret key x, and a message m, computes $S = \frac{1}{H(m)+x}P$. The signature is S.
4. Ver: Given a public key P_{pub}, a message m, and a signature S, verify whether $\hat{e}(H(m)P + P_{pub}, S) = \hat{e}(P, P)$.

The security of ZSS signature scheme is based on the security of the k-CAA problem. For more details, we refer the readers to [5].

Definition 2. k-Collusion Attack Algorithm(k-CAA)
For an integer k, and $x \in \mathbb{Z}_q$, $P \in G_1$, given $(P, Q = xP, h_1, \cdots, h_k \in \mathbb{Z}_q, \frac{1}{h_1+x}P, \cdots, \frac{1}{h_k+x}P)$, to compute $\frac{1}{h+x}P$ for some $h \notin \{h_1, h_2, \cdots, h_k\}$.

The k-CAA assumption states that for every probabilistic polynomial-time algorithm $\mathcal{A}$, $\mathsf{Succ}^{k-CCA}_{\mathcal{A},\mathbb{G}_1}$ is negligible. The following theorem has been proved in [5].

Theorem 1. *If there exists a (t, q_H, q_S, ϵ)-forger $\mathcal{F}$ using adaptive chosen message attack for the proposed signature scheme, then there exists a (t', ϵ')-algorithm $\mathcal{A}$ solving q_S-CAA, where $t' = t, \epsilon' \geq (\frac{q_S}{q_H})^{q_S}$.*

3 An Analysis of the OIO Short Proxy Signature Scheme

Recently, a short proxy signature scheme (OIO for short) was presented in [11]. In this section, we will firstly describe the OIO short proxy signature scheme, then we give an attack to show that the original signer can successfully forge a valid proxy signature of OIO's scheme.

3.1 Description of OIO Short Proxy Signature Scheme

1. Notations Used in OIO Scheme
   - $\mathcal{O}$: an original signer; $\mathcal{P}$: a proxy signer; $\mathcal{V}$: a verifier.
   - ID_p: the ID for a user p, m_p; a message to be signed by $\mathcal{P}$.
   - $\mathcal{H}(\cdot)$: a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$.
2. Key Generation:
   (a) $\mathcal{O}$ picks up two elements $P \in \mathbb{G}_1$ and $s \in Z_q$ at random and computes $V = sP$, $g = \hat{e}(P, P)$. $\mathcal{O}$ sends g to $\mathcal{P}$.
   (b) $\mathcal{P}$ picks up a random number $r \in \mathbb{Z}_q$, computes $v_p = g^r$ and sends v_p to $\mathcal{O}$. $\mathcal{O}$ then computes $e_p = \mathcal{H}(ID_p, v_p)$ and $S_p = \frac{1}{s+e_p}P$.
   (c) $\mathcal{O}$ publishes g, V, ID_p and sends S_p to $\mathcal{P}$ using a secure channel.
   (d) $\mathcal{P}$ checks whether $\hat{e}(e_pP + V, S_p) = g$ holds or not.
   As a result, $\mathcal{O}'s$ key tuple is {Public-key: g, V; Secret-key: s}. $\mathcal{P}'s$ key tuple is {Public-key: ID_p, v_p; Secret-key: r, S_p}.
3. Proxy Signature Generation: $\mathcal{P}$ computes $e_m = \mathcal{H}(m_p, v_p)$ and $Sig_p = (r + e_m)S_p$. The proxy signature for a message m_p is Sig_p.
4. Proxy Verification: $\mathcal{V}$ first computes $e_p = \mathcal{H}(ID_p, v_p)$ and $e_m = \mathcal{H}(m_p, v_p)$. Then he checks whether $\hat{e}(e_pP + V, Sig_p) = v_pg^{e_m}$ holds or not.

3.2 An Attack Model of OIO Short Proxy Signature Scheme

Suppose there is a valid message-signature pair (m, Sig_m). Since Sig_m is a valid proxy signature on the message m, we have $Sig_m = (r + e_m)S_p$. Then $\mathcal{O}$ can compute $rP = (s + e_p)Sig_m - \mathcal{H}(m, v_p)P$.

With the knowledge of rP, the original signer is able to forge a valid signature on any new message. For a new message m^*, $\mathcal{O}$ computes $Sig_{m^*} = \frac{rP+e_{m^*}P}{s+e_p}$ where $rP = (s + e_p)Sig_m - \mathcal{H}(m, v_p)P$, $e_{m^*} = \mathcal{H}(m^*, v_p)$ and $e_p = \mathcal{H}(ID_p, v_p)$ are all known to $\mathcal{O}$. We can find it is a valid proxy signature on the message m^* because $Sig_{m^*} = \frac{rP+e_{m^*}P}{s+e_p} = (r + e_{m^*})\frac{1}{s+e_p}P = (r + e_{m^*})S_p$ which is indistinguishable to the third party which party ($\mathcal{P}$ or $\mathcal{O}$) is the signer.

4 Outline of Our Short Proxy Signature (SPS) Scheme

Let Alice denote the origin signer and Bob the proxy signer. Our short proxy signature scheme consists of the following algorithms: ParamGen, KeyGen, ProxyKeyGen, ProxySign and ProxyVer.

1. ParamGen: Taking as input the system security parameter k, this algorithm outputs system's parameters: Para.
2. KeyGen: Taking as input the system security parameter k, the algorithm generates the secret/public key pair (x_i, P_i) where $i \in \{A, B\}$. That is $(x_i, P_i) \leftarrow \mathsf{KeyGen}(\mathsf{Para})$.
3. ProxKeyGen: The original signer Alice and the proxy signer Bob utilize this algorithm to obtain the proxy key which will be used in the ProxySign. That is $\mathsf{proxykey} \leftarrow \mathsf{ProxyKeyGen}(\mathsf{Para}, x_A, P_A, x_B, P_B, ID_B, m_w)$. m_w is the warrant which specifies what kinds of messages are delegated and ID_B is the identity of the proxy signer Bob.
4. ProxySign: The proxy signer utilizes this algorithm to generate the proxy signature. That is $\sigma \leftarrow \mathsf{ProxySign}(m, \mathsf{proxy\ key}, \mathsf{Para})$.
5. ProxyVer: Given the public keys of the origin signer and proxy signer, anyone can use this algorithm to check whether a signature is a valid proxy signature. That is $\{\mathsf{True}, \perp\} \leftarrow \mathsf{ProxyVer}(m, \sigma, P_A, P_B, ID_B, m_w, \mathsf{Para})$

4.1 Attack Model

To discuss the Non-Forgeability of our short proxy signature scheme, we divide the adversaries into the following three types:

1. **Type I**: The adversary only has the public keys of Alice and Bob.
2. **Type II**: The adversary has the public keys of Alice and Bob and also has the secret key of Bob.
3. **Type III**: The adversary has the public keys of Alice and Bob and also has the secret key of Alice.

One can find that if our short proxy signature scheme is unforgeable against Type II (or Type III) adversary, our scheme is also unforgeable against Type I adversary.

Formal Security Notion

Type II Adversary

We provide a formal definition of existential unforgeability of a short proxy signature scheme under a Type II chosen message attack (EF-SPS-adversary). This type of adversaries only has the secret key of the proxy signer and does not obtain the proxy key from the original signer. It is defined using the following game between an adversary $\mathcal{A}_{II}$ and a challenger $\mathcal{C}$.

- Setup: $\mathcal{C}$ runs the algorithm to obtain the secret key and public key pair (x_A, P_A), (x_B, P_B) representing the keys of the original signer A and the proxy signer B, respectively. $\mathcal{C}$ then sends (P_A, P_B, x_B) to the adversary $\mathcal{A}_{II}$.

- PublicKey Queries: $\mathcal{A}_{II}$ can set the i^{th} user in the system as the proxy signer. He asks the public key P_i of the i^{th} user with the identity ID_i. In response, $\mathcal{C}$ generates P_i and returns P_i to the adversary $\mathcal{A}_{II}$.
- PSign Queries: $\mathcal{A}_{II}$ can request a signature on a message m with the original signer A and the proxy signer with the identity ID_i. In response, $\mathcal{C}$ outputs a signature σ for a message m.
- Output: Finally, $\mathcal{A}_{II}$ outputs a target message $m^* \in \{0,1\}^*$ and σ^* such that σ^* is a valid proxy signature with the original signer A and the proxy signer B.

Type III Adversary
We provide a formal definition of existential unforgeability of a short proxy signature scheme under a Type III chosen message attack (EF-SPS-adversary). It is defined using the following game between an adversary $\mathcal{A}_{III}$ and a challenger $\mathcal{C}$.

- Setup: $\mathcal{C}$ runs the algorithm to obtain the secret key and public key pair (x_A, P_A), (x_B, P_B) representing the keys of the original signer A and the proxy signer B, respectively. $\mathcal{C}$ then sends (P_A, P_B, x_A) to the adversary $\mathcal{A}_{III}$.
- PSign Queries: $\mathcal{A}_{III}$ can request a signature on a message m. In response, $\mathcal{C}$ outputs a signature σ for a message m.
- Output: Finally, $\mathcal{A}_{III}$ outputs a target message $m^* \in \{0,1\}^*$ where m^* has never been queried during the PSign Queries and σ^* is a valid proxy signature with the original signer A and the proxy signer B.

Definition 3. *A short proxy signature scheme is existential unforgeable against chosen-message attacks iff it is secure against both type II and type III adversaries.*

5 Our Scheme

1. ParamGen: Taking as input the system security parameter k, this algorithm outputs $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P\}$, including a cyclic additive group $\mathbb{G}_1$ of order q, a multiplicative group $\mathbb{G}_2$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ and a generator P of $\mathbb{G}_1$. This algorithm also outputs two cryptographic hash functions H_0 and H_1 where $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$. We denote the set $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$ where 0 is the zero element of the field $\mathbb{Z}_q$.
2. KeyGen: The algorithm generates the original signer Alice's secret/public key pair $(x_A, P_A = x_A P)$ and the proxy signer Bob's secret/public key pair $(x_B, P_B = x_B P)$.
3. ProxyKeyGen:
   (a) Alice computes $D_{AB} = x_A Q_B$. Here, $Q_B = H_0(ID_B, P_B, m_w)$. ID_B is the identity of the proxy signer Bob, P_B is the public key of Bob, and m_w is the warrant. Alice then sends D_{AB} to Bob.
   (b) Bob verifies whether $\hat{e}(D_{AB}, P) = \hat{e}(Q_B, P_A)$.
   As a result, Bob obtains his proxy key (x_B, D_{AB}).

4. ProxySign: For a message m, Bob computes $\sigma = \frac{1}{H_1(m)+x_B} D_{AB}$. The proxy signature on the message m is σ.
5. ProxyVer: To check whether σ is a valid proxy signature, any one can check: $\hat{e}(\sigma, H_1(m)P + P_B) \stackrel{?}{=} \hat{e}(Q_B, P_A)$. If the equation holds, the receiver accepts it as a valid proxy signature; otherwise, rejects it.

The correctness of the scheme can be verified:

$$\begin{aligned}\hat{e}(\sigma, H_1(m)P + P_B) &= \hat{e}(\frac{1}{H_1(m)+x_B} D_{AB}, H_1(m)P + P_B) \\ &= \hat{e}(\frac{1}{H_1(m)+x_B} D_{AB}, (H_1(m)+x_B)P) \\ &= \hat{e}(D_{AB}, P) = \hat{e}(Q_B, P_A)\end{aligned}$$

6 Security Analysis

6.1 Unforgeable Against Type II Adversary

Theorem 2. *Let $\mathcal{A}_{II}$ be a type II adversary who can get a valid signature of our scheme with success probability $Succ^{EF-CMA}_{\mathcal{A}_{II},SPS}$. In some polynomial time t, he can ask q_H hash queries to the hash function H_1 and q_S sign queries and q_V verify queries, then there exits $\mathcal{B}$ who can use $\mathcal{A}_{II}$ to solve an instance of CDH problem with the success probability*

$$Succ^{CDH}_{\mathcal{B},\mathbb{G}_1} = Succ^{EF-CMA}_{\mathcal{A}_{II},SPS}$$

in the same polynomial time t.

Proof. Given $P_1 = aP, P_2 = bP$ for some unknown $a, b \in \mathbb{Z}_q^*$, we will show how $\mathcal{B}$ can use the type II adversary $\mathcal{A}_{II}$ to get the value abP. Let's recall the definition of the type II adversary $\mathcal{A}_{II}$. This type of adversary $\mathcal{A}_{II}$ only has the secret key of the proxy signer Bob.

$\mathcal{B}$ chooses $c \in_R \mathbb{Z}_q^*$ and sets the original signer's public key $P_A = P_1 = aP$, the proxy signer's public key $P_B = cP$ and $Q_B = P_2 = bP$. $\mathcal{B}$ returns (P, P_A, P_B, Q_B, c) to the Type II adversary $\mathcal{A}_{II}$. $\mathcal{A}_{II}$ can ask most q_H PHash Queries and q_S PSign Queries to the PHash Oracle and PSign Oracle respectively. $\mathcal{A}_{II}$ can additionally request the PublicKey Oracle of the other proxy signer's public key he is interested in. $\mathcal{B}$ will act all these oracles in our proof. After all the queries, $\mathcal{A}_{II}$ will output a valid proxy signature (m^*, σ^*) such that $\hat{e}(\sigma^*, H_1(m^*)P + P_B) = \hat{e}(Q_B, P_A)$. Here, we assume that m^* has been queried by $\mathcal{A}_{II}$ to the PHash Oracles before he outputs the signature σ^* of the message m^*.

In the proof $\mathcal{B}$ maintains a list, *H-List*, to record all the PHash Queries and the corresponding answers. $\mathcal{B}$ also maintains another list *PK-List* to record the public key queries and the corresponding answers. We assume that before $\mathcal{A}_{II}$ asks the PSign Queries with the i^{th} user is proxy signer, $\mathcal{A}_{II}$ has obtained the public key P_i and Q_i of the i^{th} proxy signer from the PublicKey Oracle.

1. Public Key Queries: In this process, $\mathcal{A}_{II}$ can ask the P_i and Q_i of the i^{th} proxy signer with the identity ID_i. For each request, $\mathcal{B}$ chooses $x_i, y_i \in \mathbb{Z}_q^*$, and sets $P_i = x_i P, Q_i = y_i P$. $\mathcal{B}$ then adds (ID_i, x_i, y_i) to the *PK-List* and returns (P_i, Q_i) to A_{II}.
2. PHash Queries: In this process, $\mathcal{A}_{II}$ can ask at most q_H PHash Queries. For each request m_i, $\mathcal{B}$ first checks the *H-List*:
   (a) If there is an item (m_j, h_j) in the *H-List* such that $m_j = m_i$, $\mathcal{B}$ sets $H_1(m_i) = h_j$ and returns h_j as the hash value of m_i to $\mathcal{A}_{II}$.
   (b) Otherwise, m_i has not been requested to the hash oracle. $\mathcal{B}$ chooses $h_i \in \mathbb{Z}_q^*$ such that there is no item $(\cdot, h_i)$ in the *H-List*. $\mathcal{B}$ then adds (m_i, h_i) into the *H-List* and returns h_i to $\mathcal{A}_{II}$.
3. PSign Queries: In this process, $\mathcal{A}_{II}$ can ask at most q_S PSign Queries. For each request (m_i, ID_k) chosen by $\mathcal{A}_{II}$, $\mathcal{B}$ first checks the *H-List*:
   (a) If there is an item (m_j, h_j) in the *H-List* such that $m_j = m_i$, $\mathcal{B}$ obtains $H_1(m_i) = h_j$.
   (b) Otherwise, m_i has not been requested to the hash oracle. $\mathcal{B}$ chooses $h_i \in \mathbb{Z}_q^*$ such that there is no item $(\cdot, h_i)$ in the *H-List*. $\mathcal{B}$ then adds (m_i, h_i) into the *H-List* and sets $H_1(m_i) = h_i$.

   After the check of *H-List*, $\mathcal{B}$ returns $\sigma_i = \frac{1}{h_i + x_k} y_k P_A$ to $\mathcal{A}_{II}$ as the signature of m_i under the original signer A and the k^{th} proxy signer.

After all the queries, $\mathcal{A}_{II}$ outputs (m^*, σ^*) such that $\hat{e}(\sigma^*, H_1(m^*)P + P_B^*) = \hat{e}(Q_B, P_A)$. That is $\sigma^* = \frac{1}{H_1(m^*)+c} abP$. Therefore, $\mathcal{B}$ computes $(H_1(m^*)+c)\sigma^* = (H_1(m^*)+c)\frac{1}{H_1(m^*)+c} abP = abP$. Therefore $\mathcal{B}$ can also solve an instance of CDH problem with the probability $Succ^{CDH}_{\mathcal{B},\mathbb{G}_1} = Succ^{EF-CMA}_{\mathcal{A}_{II},SPS}$. ∎

6.2 Unforgeable Against Type III Adversary

Theorem 3. *Let $\mathcal{A}_{III}$ be a type III adversary who can get a valid signature of our short proxy signature (SPS) scheme with probability $Succ^{EF-CMA}_{\mathcal{A}_{III},SPS}$. In polynomial time t he can aske q_H hash queries to the hash function H_1 and q_S sign queries and q_V verify queries, then there exists another adversary $\mathcal{B}$ also uses $\mathcal{A}_{III}$ to obtain a valid signature of ZSS signature scheme [5] with the success probability*

$$Succ^{EF-CMA}_{\mathcal{B},\ ZSS} = Succ^{EF-CMA}_{\mathcal{A}_{III},\ SPS}$$

in the same polynomial time t.

Proof. There two adversaries, $\mathcal{A}_{III}$ and $\mathcal{B}$ in our proof. $\mathcal{A}_{III}$ is the Type III attacker of our proposed short proxy signature(SPS) scheme and $\mathcal{B}$ is the adversary of ZSS signature scheme [5]. We will show that given Bob's public key P_B, how $\mathcal{B}$ can use $\mathcal{A}_{III}$ to obtain Bob's valid signature of ZSS scheme in [5]. As presented in [5], $\mathcal{B}$ can ask Hash Query and Sign Query to his own Hash Oracle and Sign Oracle.

In the proof, $\mathcal{A}_{III}$ can ask the PHash Query and PSign Query. $\mathcal{B}$ will act as these three oracles. $\mathcal{B}$ chooses $a, c \in \mathbb{Z}_q^*$ and sets Alice's public key $P_A = aP$ and $Q_B = cP$. Then, $\mathcal{B}$ returns P_A, P_B, Q_B, a to the adversary $\mathcal{A}_{III}$. $\mathcal{A}_{III}$ can ask the following queries:

1. PHash Queries: In this process, $\mathcal{A}_{III}$ can ask at most q_H PHash Queries. For each request m_i, $\mathcal{B}$ submits m_i to his own Hash Oracle and obtains the result h_i. $\mathcal{B}$ also returns h_i to $\mathcal{A}$ as the answer.
2. PSign Queries: In this process, $\mathcal{A}_{III}$ can ask at most q_S PSign Queries. For each request m_i, $\mathcal{B}$ submits m_i to the Sign Oracle and obtains the result $\hat{\sigma}_i$. Then $\mathcal{B}$ returns $\sigma_i = ac\hat{\sigma}_i$ to $\mathcal{A}_{III}$ as the answer. Note that σ_i is a valid proxy signature, this is true because $\hat{\sigma}_i$ is Bob's valid signature of *ZSS*, that is $\hat{e}(\hat{\sigma}_i, H_1(m_i)P + P_B) = \hat{e}(P, P)$. Therefore $\hat{e}(\sigma_i, H_1(m_i)P + P_B) = \hat{e}(ac\hat{\sigma}_i, H_1(m_i)P + P_B) = \hat{e}(\hat{\sigma}_i, H_1(m_i)P + P_B)^{ac} = \hat{e}(cP, aP) = \hat{e}(Q_B, P_A)$

After all the queries, $\mathcal{A}_{III}$ outputs (m^*, σ^*) such that m^* is not requested in the PSign Queries and $\hat{e}(\sigma^*, H_1(m^*)P + P_B) = \hat{e}(Q_B, P_A)$. Then $\mathcal{B}$ computes $\hat{\sigma^*} = (ac)^{-1}\sigma^*$ and $(m^*, \hat{\sigma^*})$ is Bob's valid signature in the scheme presented in [5]. This is true because: $\hat{e}(\hat{\sigma^*}, H_1(m^*)P + P_B) = \hat{e}(\sigma^*, H_1(m^*)P + P_B)^{(ac)^{-1}} = \hat{e}(Q_B, P_A)^{(ac)^{-1}} = \hat{e}(cP, aP)^{(ac)^{-1}} = \hat{e}(P, P)$. That is to say $\mathcal{B}$ also find a valid signature of *ZSS* signature scheme [5] with the probability $Succ^{EF-CMA}_{\mathcal{B},\, ZSS} = Succ^{EF-CMA}_{\mathcal{A}_{III},\, SPS}$ in the same polynomial time t. ∎

7 Conclusion

In this paper, firstly we pointed out that the construction of short proxy signature (OIO scheme) in [4] is insecure. We proceed with a formal definition of short proxy signature scheme, together with three types of adversarial model. Finally, we presented an efficient and short proxy signature, which outperforms any existing proxy signature in terms of signature length, and proved that the scheme is secure in the random oracle model.

Acknowledgement

The authors would like to express their gratitude thanks to the anonymous referees of the 2nd International Symposium on Ubiquitous Intelligence and Smart Worlds (UISW2005) 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,* LNCS 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,* LNCS 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,* LNCS 2248, pages 514–532. Springer–Verlag, 2001.

4. A. Boldyreva A. Palacio and B. Warinschi. Secure Proxy Signature Schemes for Delegation of Digning Rights. Available at http://eprint.iacr.org/2003/096
5. F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Public Key Cryptography (PKC'04),* LNCS 2947, pages 277–290. Springer–Verlag, 2004.
6. 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,* LNCS 2612, pages. 68–79. Springer–Verlag, 2003.
7. B. Lee, H. Kim and K. Kim. Strong proxy Signature and its applications. In *Proc of SCIS'01*, pages. 603–08. 2001.
8. B. Lee, H. Kim, and K. Kim. Secure mobile agent using strong nondesignated proxy signature. In *Information Security and Privacy (ACISP'01),* LNCS 2119, pages. 474–486. Springer–Verlag, 2001.
9. S. Kim, S. Park and D. Won. Proxy Signatures, Revisited. In *Information and Communications Security (ICICS'97),* LNCS 1334, pages. 223–232. Springer–Verlag, 1997.
10. 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.
11. 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.
12. H.-U. Park and I.-Y. Lee. A digital nominative proxy signature scheme for mobile communications. In *Information and Communications Security (ICICS'01),* LNCS 2229, pages. 451–455, Springer–Verlag, 2001.