A Novel Secure Self Proxy Signature Scheme

Samaneh Mashhadi

Department of Mathematics, Iran University of Science & Technology,
Hengam Street, Resalat Square, Narmak, Tehran16846 13114, Iran
(Email:smashhadi@iust.ac.ir)

(Received Apr. 24, 2011; revised and accepted Oct. 12, 2011)

Abstract

Recently, Kim *et al.*'s proposed a new type of proxy signature scheme, called self proxy signature scheme. In their scheme, a signer, delegates his/her signing capability to himself/herself and uses the proxy private/public key pair as temporary keys. In this paper, we will demonstrate some security leaks inherent in Kim *et al.*'s scheme and show that an adversary can forge a valid self proxy signature for any message by using different ways. Finally, we will propose an improvement to eliminate the pointed out security leaks.

Keywords: Distinguishability, self proxy signature, undeniability, unforgeability

1 Introduction

The concept of a proxy signature was first introduced by Mambo *et al.* [9] in 1996. In a proxy signature scheme, generally, there are two entities: an original signer and a proxy signer. The original signer can delegate his signing power to a proxy signer. The proxy signer can generate a valid signature on behalf of the original signer. Since then, many proxy signature schemes have been proposed [4, 6, 7, 10]. Proxy signatures can combine other special signatures to obtain some new types of proxy signatures. Up to now, various kinds of proxy signature scheme have been proposed, such as threshold proxy signature [2], proxy blind signature [1], multi-proxy signature [3, 8], etc.

Recently, the concept of self proxy signature scheme was introduced by Kim *et al.* [5]. In a self proxy signature scheme a signer, Alice delegates her signing capability to herself recursively. Using this scheme, Alice generates many proxy private/public key pairs, uses them simultaneously and revokes the temporary keys easily. Furthermore, it is easy to revoke the temporary private/public key pair.

A self proxy signature scheme is a useful tool in the real world. For example, a person may use a legal seal and many other seals simultaneously. After registering, the legal seal is used in an important work, and the other seals are used for normal works. To use seals like this, the person protects the legal seal and uses another one for only a particular work.

A secure self proxy signature scheme should satisfy the following requirements [5]: unforgeability, undeniability and distinguishability. Unforgeability requirement is to ensure that only the valid signer can create the self proxy signature. Undeniability property means that the signer cannot deny his/her signatures to anyone. Distinguishability property means that the self proxy signature must be distinguishable from the normal signature.

In this paper, we will show that Kim *et al.*'s scheme cannot achieve the security requirements. To remedy these weaknesses, we propose a new improvement. The rest of this paper is sketched as follows: in Section 2, we will review Kim *et al.*'s scheme. In Section 3, we will point out the security leaks inherent in Kim *et al.*'s scheme. A novel improvement is proposed in Section 4. In Section 5 we analyze the security of our scheme. Finally, we give conclusions in Section 6.

2 Brief Review of Kim *et al.*'s Scheme

In this section, we briefly review Kim *et al.*'s scheme [5]. The scheme consists of three phases: key generation, signature generation and verification phases.

Throughout the paper, p denotes a large prime, q is a prime divisor of p-1, g is a generator in $\mathbb{Z}_p^*$ with order q, and $H(\cdot)$ denotes a public cryptographically strong hash function. Also m denotes the message to be signed and m_W is a warrant, specifying the delegation period for m, the identities of the signer, etc. Further, Alice is always the signer and Bob is always the verifier.

Alice selects a private key $x_a \in \mathbb{Z}_q^*$, and obtains a public key $y_a = g^{x_a} \bmod p$.

2.1 Self Proxy Key Generation Phase

Alice generates the temporary self proxy private/public key pair by using her original key pair (x_a, y_a) as follows:

She chooses random numbers k, $x_t \in \mathbb{Z}_q^*$, and computes

$$r = g^k \bmod p, \quad y_t = g^{x_t} \bmod p.$$

Then she computes self proxy private key as

$$x_p = k + (x_a + x_t)H(m_W) \bmod q,$$

and obtains its corresponding self proxy public key $y_p = g^{x_p} \bmod p$. Finally she publishes y_t.

2.2 Self Proxy Signature Generation Phase

Suppose that Alice want to generate a self proxy signature of a message m. She performs the following operations.

She chooses a random number $k' \in \mathbb{Z}_q^*$, and computes r' and S' as follows:

$$\begin{aligned} r' &= g^{k'} \bmod p, \\ S' &= k' + x_p h(m) \bmod q. \end{aligned}$$

Then she sends $(m, (r', S'), r, m_W)$ to Bob.

2.3 Self Proxy Signature Verification Phase

First of all, Bob checks the signer's identity and the delegation lifetime of the warrant m_W. If all validations hold, he recovers the self proxy public key y_p as follows:

$$y_p = r(y_a y_t)^{H(m_W)} \bmod p,$$

and checks the validity of the next equality.

$$g^{S'} = r' y_p^{H(m)} \bmod p.$$

If the equality holds, he accepts $(m, (r', S'), r, m_W)$ as the valid self proxy signature.

3 Attack on Kim *et al.*'s Scheme

Kim *et al.* claimed that their scheme was secure. However, in this section, we shall show that their scheme cannot achieve their claimed security requirements. From now on without loss of generality, we assume that Oskar is an adversary.

3.1 Unforgeability property

Here, we show that Kim *et al.*'s scheme cannot satisfy the unforgeability property. Indeed, we show that an adversary can forge self proxy signatures by using one of the following three ways:

3.1.1 Attack 1

We show that after intercepting a valid self proxy signature generated by Alice, Oskar can forge a self proxy signature.

Suppose after intercepting a valid self proxy signature $(m, (r', S'), r, m_W)$, Oskar wants to forge a valid self proxy signature for any message m_1. For this purpose, he chooses a random number $k_1 \in \mathbb{Z}_q^*$ and computes

$$r_1 = g^{k_1} y_p^{-H(m_1)} \bmod p, S_1 = k_1 \bmod q.$$

Now, $(m_1, (r_1, S_1), r, m_W)$ is the valid self proxy signature. This is because

$$\begin{aligned} r_1 y_p^{H(m_1)} &= g^{k_1} y_p^{-H(m_1)} y_p^{H(m_1)} \\ &= g^{S_1} \bmod p. \end{aligned}$$

Therefore, Alice has never signed the message m_1, but she cannot deny.

3.1.2 Attack 2

In the following, we show that Oskar can forge a self proxy signature with his desired warrant on behalf of Alice, by changing Alice's self proxy public key.

For this purpose, he chooses random integers α, $\beta \in \mathbb{Z}_q^*$, forges the desired warrant m'_W which records Alice's identity. Then he computes

$$\begin{aligned} r_1 &= g^{\alpha} (y_a y_t)^{-H(m'_W)} \bmod p, \\ r'_1 &= g^{\beta} \bmod p. \end{aligned}$$

Next he computes

$$S_1 = \beta + \alpha H(m_1) \bmod q.$$

Now, $(m_1, (r'_1, S_1), r_1, m'_W)$ is the valid self proxy signature of the message m_1. We give the reasons as follows:

Bob, recovers the self proxy public key y'_p as follows:

$$\begin{aligned} y'_p &= r_1 (y_a y_t)^{H(m'_W)} \\ &= g^{\alpha} (y_a y_t)^{-H(m'_W)} (y_a y_t)^{H(m'_W)} \\ &= g^{\alpha} \bmod p, \end{aligned}$$

and checks the validity of the next equality.

$$g^{S_1} = g^{\beta + \alpha H(m_1)} = r'_1 y_p'^{H(m_1)} \bmod p.$$

That is, Oskar can forge a valid self proxy signature by changing the content of warrant and temporary public key.

3.2 Undeniability Property

From what have been analyzed in the Subsection 3.1., Oskar can forge a valid proxy signature for any message by using different ways and claim dishonestly that has been generated by Alice. Therefore, a malicious signer, Alice can generate a valid self proxy signature and claim that has been generated by an attacker. In fact, Alice can sign the message m, and deny her signature. Therefore, Kim *et al.*'s scheme cannot satisfy the undeniability property.

4 The Proposed Scheme

To resist the attacks pointed out in the previous section, we propose an improved scheme. Our scheme can be divided into four phases: the setup, key generation, signature generation and verification phases. The system parameters are the same as those in Kim *et al.*'s scheme.

4.1 Setup Phase

Alice generates the pair $(x_a,\, y_a)$ of private key x_a and public key y_a as follows:

She randomly chooses $x_a \in \mathbb{Z}_q^*$, computes $y_a{=}g^{x_a}$ mod p, and sends y_a to the certificate authority (CA). Then CA randomly chooses $x_0 \in \mathbb{Z}_q^*$, computes $A{=}g^{x_0}$mod$p$, and returns A to Alice. Next Alice computes $A_a{=}A^{x_a}$mod$p$, and sends A_a to CA. The certificate authority checks the equality $y_a^{x_0} = A_a$; if it holds, CA accepts their certification, otherwise he refuses it.

4.2 Self Proxy Key Generation Phase

Alice generates the temporary self proxy private/public key pair by using her original key pair $(x_a,\, y_a)$ as follows:

She chooses random numbers $k,\, x_t \in \mathbb{Z}_q^*$, and computes

$$\begin{aligned} r &= g^k \bmod p, \\ y_t &= g^{x_t} \bmod p. \end{aligned}$$

Then she computes self proxy private key as

$$x_p = k + (x_a + x_t)H(m_W)r \bmod q,$$

and obtains its corresponding self proxy public key $y_p = g^{x_p} \bmod p$. Finally she publishes y_t.

4.3 Self Proxy Signature Generation Phase

Suppose that Alice want to generate a self proxy signature of a message m. She performs the following operations.

She chooses a random number $k' \in \mathbb{Z}_q^*$, and computes r' and S' as follows:

$$\begin{aligned} r' &= g^{k'} \bmod p, \\ S' &= k' + x_p h(m, m_W)r' \bmod q. \end{aligned}$$

Then she sends $(m, (r', S'), r, m_W)$ to Bob.

4.4 Self Proxy Signature Verification Phase

$$y_p = r(y_a y_t)^{H(m_W)r} \bmod p,$$

and checks the validity of the next equality.

$$g^{S'} = r' y_p^{H(m,m_W)r'} \bmod p.$$

If the equality holds, he accepts $(m, (r', S'), r, m_W)$ as the valid self proxy signature.

5 Discussions

In this section, we show that our proposed scheme withstands the attacks mentioned in Section 4. Furthermore, we show that the requirements of a self proxy signature scheme are fulfilled in our scheme. Finally, in terms of computational complexity, we compare the new self proxy signature with Kim *et al.*'s scheme.

5.1 Attack 1

We will show that after intercepting a valid self proxy signature $(m, (r', S'), r, m_W)$, Oskar cannot forge a valid self proxy signature for a message m_1.

Suppose Oskar wants to forge a self proxy signature $(m_1, (r_1, S_1), r, m_W)$ for a message m_1, and claim dishonestly that has been generated by Alice. For this purpose, he chooses random integers $\alpha \in \mathbb{Z}_q^*$. Now, he should calculate r_1 and S_1 as follows: $r_1 = g^{\alpha} y_p^{-H(m_1,m_W)r_1} \bmod p$, $S_1 = \alpha \bmod q$.

However, he should solve the discrete logarithm problem $y_p = g^{x_p} \bmod p$ in order to compute the above r_1. Therefore, no adversary can forge a valid self proxy signature by this attack.

5.2 Attack 2

Suppose Oskar tries to forge a valid proxy signature $(m_1, (r_1', S_1), r_1, m_W')$. For this purpose, he can generate the desired warrant m_W' and choose random integers $\alpha, \beta \in \mathbb{Z}_q^*$, and then compute $r_1' = g^{\beta} \bmod p$. Now, Oskar should calculate r_1, S_1 as follow:

$$\begin{aligned} r_1 &= g^{\alpha}(y_a y_t)^{-H(m_W')r_1} \bmod p, \\ S_1 &= \beta + \alpha H(m_1, m_W')r_1' \bmod q. \end{aligned}$$

However, he should solve the discrete logarithm problem $y_a y_t = g^{x_p + x_t} \bmod p$ in order to compute the above r_1. Therefore, no adversary can forge a valid self proxy signature by this attack.

5.3 Security Properties

In this subsection, we show that the requirements of a self threshold proxy signature scheme are fulfilled in our scheme.

5.3.1 Unforgeability Property

Based on discrete logarithm problem, it is virtually impossible to obtain Alice's self proxy private key x_p from the corresponding public key y_p. Also, according to the Schnorr signature scheme, it is very difficult for anyone to obtain x_p from the self proxy signature S'. Hence, x_p can be kept secretly and be reused.

Therefore, Oskar has to forge the valid signature $(m, (r', S'), r, m_W)$ on the message m without the private key x_p by using the previous attacks. But, we have shown that all attacks fail on our scheme. Consequently, it is

Table 1: Security comparison

Security features	Kim *et al.*'s scheme	Our scheme
Undeniability	No	Yes
Unforgeability	No	Yes
Distinguishability	Yes	Yes
Time constraint	No	Yes
Known signer	No	Yes
Resist public-key substitute attacks	No	Yes
Resist warrant attacks	No	Yes

Table 2: Comparison of computational complexity

Schemes	Key generation	Signature generation	Verification
Kim *et al.*'s scheme	$2T_e + T_m + T_h$	$T_e + T_m + T_h$	$3T_e + 3T_m + 2T_h$
Our scheme	$2T_e + 2T_m + T_h$	$T_e + 2T_m + T_h$	$3T_e + 5T_m + 2T_h$

computationally difficult for Oskar to forge the self proxy signature. Therefore, the proposed scheme satisfies the unforgeability property.

5.3.2 Undeniability Property

In the proposed scheme, any valid self proxy signature $(m, (r', S'), r, m_W)$ of a message m should be generated by Alice. This is because only Alice has the self proxy private key x_p. Moreover, the warrant m_W and temporary self proxy public key y_p are created by Alice and no adversary can change them. When the self proxy signature $(m, (r', S'), r, m_W)$ is verified, the warrant m_W is checked, and the public key of signer, y_a, the temporary self proxy public key, y_p, and the public information y_t are used in the verification phase. Thus, Alice cannot deny signing the self proxy signature. Therefore, the proposed scheme satisfies the undeniability property.

5.3.3 Distinguishability Property

In the proposed scheme, when the self proxy signature $(m, (r', S'), r, m_W)$ is verified, Alice's public key and her identity are used in the verification phase; therefore, we can consider it as a self proxy signature and not a normal signature. Thus, anyone can distinguish the self proxy signature from normal signatures. Thus the proposed scheme satisfies the distinguishability property.

From what have been analyzed above, we are fully certain that the necessary requirements a self proxy signature scheme are fulfilled in our scheme.

5.4 Performance

In this section, in terms of computational complexity, we compare our scheme with Kim *et al.*'s scheme and summarize the result in Table 2. For convenience, the following notations are used to analyze the computational complexity.

- T_e: Time for one exponentiation computation.
- T_m: Time for one modular multiplication computation.
- T_h: Time for hash function computation.

6 Conclusions

In this paper, we have pointed out some security leaks of Kim *et al.*'s scheme and further proposed a novel scheme, which not only keeps the merits of the previous scheme, but also remedies the security weaknesses.

Acknowledgement

This research was partially supported by the Tafresh University, under contract no. 65401113.

References

[1] A. K. Awasthi and S. Lal, "Proxy blind signature scheme," *Trans Cryptol*, vol. 2, no. 1, pp. 5-11, 2005.

[2] H. Huanga and C. Chang, "A novel efficient(t,n) threshold proxy signature scheme," *Information Sciences*, vol. 176, no. 10, pp. 1338-1349, 2006.

[3] S. Hwang and C. Shi, "A simple multi-proxy signature scheme," *Proceedings of the 10th national conference on information security*, pp. 134-138, Hualien, Taiwan, ROC, 2000.

[4] S. Kim, S. Park, and D. Won, "Proxy signatures," *Proceedings of international conference on information and communications security (ICICS)'97*, LNCS 1334, pp. 223-232, Springer-Verlag, 1997.

[5] Y. Kim and J. Chang, "Self proxy signature scheme," *International Journal of Computer Science and Network Security*, vol. 7, pp. 335-338, 2007.

[6] B. Lee, H. Kim, and K. Kim, "Strong proxy signature and its applications," *SCIS2001*, vol. 2, no. 2, pp. 603-608, 2001.

[7] J. Lee, J. Cheon, and S. Kim, "An analysis of proxy signatures: Is a secure channel necessary," *Cryptology-CT-RSA'03,* LNCS 2612, pp. 68-79, Springer-Verlag, 2003.

[8] X. Li, K. Chen, and S. Li, "Multi-proxy signature and proxy multi-signature schemes from bilinear pairings," *Proceedings of PDCAT 2004*, LNCS 3320, pp. 591-595, Springer-Verlag, 2004.

[9] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: delegation of the power to sign messages," *Transactions on Fundamentals of Electronic Communications and Computer Science*, vol. E79-A, pp. 1338-1354, 1996.

[10] S. F. Tzeng, M. S. Hwang, and C. Y. Yang, "An improvement of nonrepudiable threshold proxy signature scheme with known signers," *Computers & Security*, vol. 23, pp. 174-178, 2004.

Samaneh Mashhadi was born in Tafresh, Iran, on March 27, 1982. She received the B.Sc. and M.Sc. degrees with honors in Mathematics from Iran University of Science and Technology (IUST), and Amirkabir University of Technology (AUT) in 2003 and 2005, respectively. She received her Ph.D. with honors in Mathematics (Cryptography) in 2008 from IUST. She is currently an Assistant Professor in Department of Mathematics of IUST. She is a member of IMS as well. Her research interests include the analysis, design, and application of digital signatures, secret sharing schemes, and security protocols etc.