Security Analysis of Some Proxy Signatures

Guilin Wang, Feng Bao, Jianying Zhou, and Robert H. Deng

Infocomm Security Department
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
http://www.i2r.a-star.edu.sg/icsd/
{glwang, baofeng, jyzhou, deng}@i2r.a-star.edu.sg

Abstract. 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. Such schemes have been suggested for use in a number of applications, particularly in distributed computing where delegation of rights is quite common. Followed by the first schemes introduced by Mambo, Usuda and Okamoto in 1996, a number of new schemes and improvements have been proposed. In this paper, we present a security analysis of four such schemes newly proposed in [15, 16]. By successfully identifying several interesting forgery attacks, we show that all the four schemes are *insecure*. Consequently, the fully distributed proxy scheme in [11] is also insecure since it is based on the (insecure) LKK scheme [14, 15]. In addition, we point out the reasons why the security proofs provided in [15] are invalid.

Keywords: digital signatures, proxy signatures, security analysis.

1 Introduction

Background. A proxy signature scheme allows one user Alice, called *original signer*, to delegate her signing capability to another user Bob, called *proxy signer*. After that, the proxy signer Bob can sign messages on behalf of the original signer Alice. Upon receiving a proxy signature on some message, a verifier can validate its correctness by the given verification procedure, and then is convinced of the original signer's agreement on the signed message. In other words, proxy signatures can be distinguished from standard signatures signed by either the original signer or the proxy signer. Proxy signature schemes have been suggested for use in a number of applications, particularly in distributed computing where delegation of rights is quite common. Examples include e-cash systems [22], mobile agents for electronic commerce [14, 15], mobile communications [23], grid computing [8], global distribution networks [3], and distributed shared object systems [18].

The basic idea of most existing proxy signature schemes is as follows. The original signer Alice sends a specific message with its signature to the proxy signer Bob, who then uses this information to construct a proxy private key. With the proxy private key, Bob can generate proxy signatures by employing a specified standard signature scheme. When a proxy signature is given, a verifier first computes the proxy public key from some public information, and then checks its validity according to the corresponding standard signature verification procedure.

Classification. Mambo, Usuda, and Okamoto introduced the concept of proxy signatures and proposed several constructions in [19]. Based on the delegation type, they classified proxy signatures as *full delegation*, *partial delegation*, and *delegation by warrant*. In full delegation, Alice's private key is given to Bob so Bob has the same signing capability as Alice. For most of real-world settings, such schemes are obviously impractical and insecure. In a partial delegation scheme, a proxy signer has a new key, called *proxy private key*, which is different from Alice's private key. So, proxy signatures generated by using proxy private key are different from Alice's standard signatures. However, the proxy signer is not limited on the range of messages he can sign. This weakness is eliminated in delegation by warrant schemes by adding a warrant that specifies what kinds of messages are delegated, and may contain other information, such as the identities of Alice and Bob, the delegation period, etc.

According to another criterion, i.e., whether the original signer knows the proxy private key, proxy signatures can also be classified as *proxy-unprotected* and *proxy-protected* schemes. This differentiation is important in practical applications, since it enables proxy signature schemes to avoid potential disputes between the original signer and proxy signer. That is, in a proxy-protected scheme the original signer cannot first generate a proxy signature, and later claims that the signature is generated by the proxy signer, not by herself. Similarly, the original signer also cannot be misattributed by the proxy signer. Since they clearly distinguish the rights and responsibilities between the original signer and the proxy signer, the proxy-protected partial delegation by warrant schemes have attracted much more investigations than others. In fact, for simplicity, this special kind of schemes is often called as proxy signature scheme.

Security Requirements. The security requirements for proxy signature are first specified in [19, 20], and later are kept almost the same beside being enhanced by [14, 15]. That is, a secure proxy signature scheme should satisfy the following five requirements:

- R1) *Verifiability*: From the proxy signature, a verifier can be convinced of the original signer's agreement on the signed message.
- R2) *Strong unforgeability*: Only the designated proxy signer can create a valid proxy signature on behalf of the original signer. In other words, the original signer and other third parties who are not designated as a proxy signer cannot create a valid proxy signature.
- R3) *Strong identifiability*: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.
- R4) *Strong undeniability*: Once a proxy signer creates a valid proxy signature on behalf of an original signer, he cannot repudiate the signature creation against anyone else.
- R5) *Prevention of misuse*: The proxy signer cannot use the proxy key for purposes other than generating a valid proxy signature. In case of misuse, the responsibility of the proxy signer should be determined explicitly.

Related Work. Followed by the first constructions given in [19, 20], a number of new schemes and improvements have been proposed [13, 32, 33, 17, 9, 22, 14, 15, 23, 27, 16, 11, 31, 5]; however, most of them do not fully meet the above listed security requirements. In [13], Kim, Park and Won introduced the concept of partial delegation by warrant, and proposed a threshold proxy signature, in which the original signing power is shared among a delegated group of n proxy singers such that only t or more of them can generate proxy signatures cooperatively. In [17], Lee et al. pointed out some weaknesses in Zhang's threshold proxy signatures [32, 33]. Later, some different opinions on their attacks are commented in [9]. In [14], Lee, Kim and Kim proposed non-designated proxy signature in which a warrant does not designate the identity of a proxy signer so any possible proxy signer can respond this delegation and become a proxy signer. Furthermore, their scheme is used to design secure mobile agents in electronic commerce setting [15]. One-time proxy signatures are studied in [2, 31].

In [16], Lee, Cheon, and Kim investigated whether a secure channel for delivery of a signed warrant is necessary in existing schemes. Their results show that if the secure channel is not provided, the MUO scheme [19] and the LKK scheme [14, 15] all are insecure. To remove the requirement of a secure channel and overcome some other weaknesses, they revised the MUO and LKK schemes. However, we will show that their revised schemes are still insecure. In addition, the PH scheme [24], which uses a 3-pass blind signature protocol to deliver the proxy private key, is also insecure [14, 16]. Boldyreva, Palacio, and Warinschi presented the formal definition and security notion for proxy signature in [5], i.e., the existential unforgeablity against adaptive chosen-message attacks [10]. At the same time, they proposed a provable secure scheme, called triple Schnorr proxy signature scheme, which is modified from the KPW scheme [13].

In contrast to the above mentioned schemes, which all are based on discrete logarithm cryptosystems, several RSA-based proxy signature schemes are proposed in [22, 15, 27]. In particular, the OTO scheme [22] and the LKK RSA-based scheme [15] are proved as secure as RSA signatures in the sense of polynomial-time reducibility. However, the OTO scheme in fact has a weak security since it is designed as a proxy-unprotected scheme; the security of Shao's schemes [27] is not proved. Moreover, the security proofs for the LKK schemes are incorrect (we will show later).

Our Contribution. In this paper, we present a security analysis of four proxy signature schemes newly proposed in [15, 16]. More specifically, we focus on the revised MUO scheme proposed by [16], the LKK scheme [14, 15] and its revised version [16], and the LKK RSA-based proxy signature proposed in [15]. To show the insecurity of these schemes, we successfully identify several forgery attacks which have different strengths and can be used in different settings. Moreover, we point out the reasons why the security proofs provided in [15] are invalid. In addition, the fully distributed proxy signature scheme in [11] is consequently broken, because it is based on the insecure LKK scheme. Note that due Lim et al.'s work [16], the authors of [11] have known their fully distributed proxy signature scheme is secure [12]. However, we will show that Lim et al.'s attack can be eliminated using a simple method explained in Section 3.1.

Organization. In Section 2, the revised MUO scheme proposed by [16] is analyzed. Then, we analyze the security of the LKK schemes [15] and the revised LKK scheme [16] in Section 3. The security analysis of the RSA-based LKK scheme [15] is given in Section 4. Finally, we conclude our paper in Section 5.

2 The MUO Scheme and Its Derivatives

Throughout this paper, p, q are two large primes such that $q|(p-1)$ and $G_q = \langle g \rangle$ is a q-order multiplicative subgroup of $\mathbb{Z}_p^*$ generated by an element $g \in \mathbb{Z}_p^*$. The discrete logarithm problem in G_q is assumed to be difficult. Hereafter, we call three such integers (p, q, g) a DLP-triple. Let $h(\cdot)$ be a collision resistant hash function. In addition, it is assumed that Alice is the original signer with a *certified key pair* (x_A, y_A) where $y_A = g^{x_A} \bmod p$, and Bob is a proxy signer with a certified key pair (x_B, y_B) where $y_B = g^{x_B} \bmod p$. Here, a certified key pair (x_A, y_A) means that Alice knows the private key x_A and has to prove her knowledge of x_A when she registers her public key certificate with a CA. In fact, this is the recommended practice for certification authorities [1, 21], and can be used to prevent rogue-key attacks [5]. We denote by m_w the *warrant* which specifies the delegation period, what kind of message m is delegated, and the identity information of the original signer (and the proxy signer), etc.

2.1 The MUO Scheme and Its Security

Proxy Key Generation: Bob gets the proxy key pair (x_P, y_P) as follows.

1) The original signer Alice generates a random number $k \in \mathbb{Z}_q^*$ and computes $K = g^k \bmod p$. Then, she calculates $s_A = x_A + k \cdot K \bmod q$, and sends (s_A, K) to Bob through a secure channel.
2) Bob checks whether $g^{s_A} \equiv y_A \cdot K^K \bmod p$. If it is, Bob computes his proxy private key x_P as $x_P = s_A + x_B \cdot y_B \bmod q$.

Proxy Signature Generation: When the proxy singer Bob signs a document m on behalf of the original signer Alice, he executes one DLP-based ordinary signing algorithm [6, 28, 7] with the proxy private key x_P. The resulting proxy signature σ is denoted by $\sigma = (Sign(m, x_P), K, y_A, y_B)$.

Proxy Signature Verification: To verify the proxy signature σ, a verifier first computes the proxy public key y_P by

$$y_P = y_A \cdot K^K \cdot y_B^{y_B} \bmod p \ (= g^{x_P} \bmod p).$$

The verification is then carried out by the same checking operations as in the corresponding standard DLP signature scheme.

In this scheme, requirement R5 is not satisfied since the warrant (s_A, K) includes neither the identity information of the proxy signer nor the limit on delegated messages. Furthermore, the secure channel for delivery the warrant is necessary. Otherwise, an interesting attack demonstrated in [16] can be mounted if Schnorr signature scheme [28] is employed to generate proxy signatures. This attack enables an attacker to convert a proxy signature σ into a new one $\bar{\sigma}$, where $\bar{\sigma}$ is also a valid proxy signature but on behalf of a different original signer. We call this attack *Original Signer Changing Attack.*

2.2 Revised MUO Scheme and Its Security

In [16], to remove the secure channel while foiling the original signer changing attack, Lee, Cheon, and Kim proposed the following revised MUO scheme. However, their revised scheme is still insecure, and in fact more vulnerable than the original MUO scheme. Since proxy signature generation and verification only involve some standard operations, we focus on the proxy key generation protocol only.

Revised Proxy Key Generation: Bob gets a proxy key pair (x_P, y_P) through the following two steps.

1) An original signer Alice generates a random number $k \in \mathbb{Z}_q^*$ and computes $K = g^k \bmod p$. After that, she calculates $s_A = x_A + k \cdot y_B \bmod q$, and then sends (s_A, K) to the proxy signer Bob (through a public, i.e., insecure, channel).
2) Bob checks whether $g^{s_A} \equiv y_A \cdot K^{y_B} \bmod p$. If (s_A, K) pass this validation, Bob computes the proxy private key as $x_P = s_A + x_B \cdot y_A \bmod q$.

The proxy public key y_P, which is used in the proxy signature verification stage, is generated as follows:

$$y_P = y_A \cdot K^{y_B} \cdot y_B^{y_A} \bmod p \; (= g^{x_P} \bmod p). \tag{1}$$

Now we discuss the security of the revised MUO scheme. We present three attacks to show that the revised MUO scheme *does not* satisfy R2, i.e., strong unforgeablility (and then does not satisfy R4, i.e., strong undeniability). In the first attack, the original signer can forge a valid proxy signature under the name of a proxy singer; our second attack allows an attacker to mount *impersonating attack*, i.e., he can generate valid proxy signature on any message by directly forging a valid proxy key pair; and in the third attack, a third party could frame Alice and Bob that they generated a proxy signature on an arbitrary message, but in fact they did not. In all three attacks, we assume that $y_B^{-1} \bmod q$ exists, i.e., $\gcd(y_B, q) = 1$. This assumption is reasonable since it holds for the overwhelming many of DLP public keys.

Forgery by the Original Signer. Suppose that the original signer Alice wants to generate valid proxy signatures which look as if they are generated by a proxy signer Bob. For this purpose, Alice does the following:

(1) Compute $b = -y_A \cdot y_B^{-1} \bmod q$ that satisfies $y_A + b \cdot y_B = 0 \bmod q$.
(2) Choose a random number $c \in \mathbb{Z}_q^*$, and define $\bar{K} = y_B^b \cdot g^c \bmod p$.
(3) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is given by:

$$\bar{x}_P = x_A + c \cdot y_B \bmod q, \quad \text{and} \quad \bar{y}_P = g^{\bar{x}_P} \bmod p. \tag{2}$$

The forged $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$ is a valid proxy key pair since it satisfies equation (1), i.e.,

$$\begin{aligned} y_A \cdot \bar{K}^{y_B} \cdot y_B^{y_A} \bmod p &= y_A \cdot (y_B^b \cdot g^c)^{y_B} \cdot y_B^{y_A} \bmod p \\ &= g^{x_A + c \cdot y_B} \cdot y_B^{y_A + b \cdot y_B} \bmod p \\ &= g^{\bar{x}_P} \bmod p \\ &= \bar{y}_P. \end{aligned}$$

Using the above forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$, Alice can generate valid proxy signature on any message m.

Impersonating Attack. In this scenario, we assume that Bob is not designated as a proxy signer by the original signer Alice. However, our following attack enables Bob to become a proxy signer of Alice by forging a valid proxy key pair $(\bar{x}_P, \bar{y}_P)$, without Alice's agreement and authorization. For this sake, he operates as follows.

(1) Compute $a = -y_B^{-1} \bmod q$ that satisfies $a \cdot y_B + 1 = 0 \bmod q$.
(2) Choose a random number $c \in \mathbb{Z}_q^*$, and define $\bar{K} = y_A^a \cdot g^c \bmod p$.
(3) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is given by

$$\bar{x}_P = c \cdot y_B + x_B \cdot y_A \bmod q, \quad \text{and} \quad \bar{y}_P = g^{\bar{x}_P} \bmod p. \tag{3}$$

Similarly, the validity of the forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$ can be verified, i.e., $\bar{y}_P = g^{\bar{x}_P} \bmod p = y_A \cdot \bar{K}^{y_B} \cdot y_B^{y_A} \bmod p$. Using this forged but valid $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$, Bob can generate valid proxy signatures for arbitrary messages of his choice. In addition, by defining $\bar{s}_A = c \cdot y_B \bmod q$, we have $g^{\bar{s}_A} = y_A \cdot \bar{K}^{y_B} \bmod p$. Therefore, different from the previous attack, in this attack Bob can provide $(\bar{s}_A, \bar{K})$ as a proof to show that he " indeed" is a proxy signer designated by the original signer Alice. It is easy to see that the distributions of $(\bar{s}_A, \bar{K}, \bar{x}_P, \bar{y}_B)$ and (s_A, K, x_P, y_B) are perfectly statistically indistinguishable. Thus, a third party cannot tell whether $(\bar{s}_A, \bar{K}, \bar{x}_P, \bar{y}_B)$ is a tuple generated normally. Our attack also means that in the revised MUO scheme, any user, if he likes, "automatically" becomes a proxy singer of another user.

Framing Attack. In this attack, a third party Charlie can forge a proxy private key $\bar{x}_P$ and then generate valid proxy signatures such that a verifier believes that these proxy signatures were signed by the proxy signer Bob on behalf of the original signer Alice. To accomplish the attack, Charlie does the followings.

(1) Compute $a = -y_B^{-1} \bmod q$, and $b = -y_A \cdot y_B^{-1} \bmod q$.
(2) Select a random number $c \in \mathbb{Z}_q^*$, and define $\bar{K} = y_A^a \cdot y_B^b \cdot g^c \bmod p$.
(3) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is computed from

$$\bar{x}_P = c \cdot y_B \bmod q, \quad \text{and} \quad \bar{y}_P = g^{\bar{x}_P} \bmod p. \tag{4}$$

Again, it can be verified directly that the forged $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$ is a valid proxy key pair, i.e., equation (1) holds.

After getting the above forged but valid proxy key pair $(\bar{x}_P, \bar{y}_P)$, Charlie can select an arbitrary message m which is beneficial to himself. Then, by using the proxy private key $\bar{x}_P$ and $\bar{K}$, he generates a proxy signature $\bar{\sigma} = (Sign(m, \bar{x}_P), \bar{K}, y_A, y_B)$. Since $(\bar{x}_P, \bar{y}_P)$ and $\bar{K}$ satisfy the proxy public key verification equation (1), this forged proxy signature will pass the standard DLP-based signature verification equation respect to the proxy public key $\bar{y}_P$. When such a forged proxy signature is presented, Alice and Bob cannot deny that $\bar{\sigma}$ is not Bob's proxy signature on behalf of Alice. Therefore, the result is that Alice and Bob will be framed. And the worst is that they cannot defend for themselves, because a judge cannot tell whether they are framed or they collude together to cheat the verifier.

In addition, we would like to point out that the revised MUO scheme has other two weaknesses. The first one is that if (s_A, K) is a warrant for Bob, $(\bar{s}_A, \bar{K})$ is also a valid warrant for Bob, where $\bar{K} = K \cdot g^{\bar{k}} \bmod p$, $\bar{s}_A = s_A + \bar{k} \cdot y_B \bmod q$, and $\bar{k} \in_R \mathbb{Z}_q^*$ is any random number. In addition, from Bob's warrant (s_A, K) (remember in essence (s_A, K) is public information), Charlie can derive a new warrant $(\bar{s}_A, \bar{K})$ for himself, if q does not divide his public key y_C. That is, he first computes $l = y_B \cdot y_C^{-1} \bmod q$, and selects a random number $\bar{k} \in \mathbb{Z}_q^*$. Then, Charlie defines $\bar{s}_A = s_A + \bar{k} \cdot y_C \bmod q$ ($= x_A + (kl + \bar{k}) \cdot y_C \bmod q$), and $\bar{K} = K^l \cdot g^{\bar{k}} \bmod p$. Since $g^{\bar{s}_A} = y_A \cdot \bar{K}^{y_C} \bmod p$, Charlie gets his proxy private key $\bar{x}_P$ by $\bar{x}_P = \bar{s}_A + x_C \cdot y_A \bmod q$.

3 The LKK Scheme and Its Derivatives

3.1 The LKK Scheme and Its Security

Proxy Key Generation: The original signer Alice uses the Schonrr scheme [28] to sign a warrant m_w, which specifies which messages Bob can sign on behalf of Alice, the validity period of delegation, etc. That is, Alice chooses a random number $k_A \in \mathbb{Z}_q^*$, computes $r_A = g^{k_A} \bmod p$ and $s_A = k_A + x_A \cdot h(m_w, r_A) \bmod q$. Then, the tuple (m_w, r_A, s_A) is sent to the proxy signer Bob, and Bob checks its validity by

$$g^{s_A} \equiv y_A^{h(m_w, r_A)} \cdot r_A \bmod p.$$

If this verification is correct, Bob sets his proxy key pair (x_P, y_P) as

$$x_P = s_A + x_B \bmod q, \quad y_P = g^{x_P} \bmod p \ (= y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \bmod p).$$

Proxy Signature Generation: With the proxy key pair (x_P, y_P), Bob can use any DLP-based signature scheme to generate proxy signature on any delegated message m. The resulting proxy signature is a tuple $\sigma = (sign(m, x_P), m_w, r_A, y_A, y_B)$.
Proxy Signature Verification: To check the validity of σ, a verifier first checks whether message m conforms to the warrant m_w. If this check is positive, he computes the proxy public key y_P as follows:

$$y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \bmod p \ (= g^{x_P} \bmod p). \tag{5}$$

Finally, the verifier checks whether $sign(m, x_P)$ is a valid signature on message m with respect to the proxy public key y_P in the corresponding DLP-based signature scheme.

The authors of [15] proved that the (Schonrr-based) LKK scheme is as secure as the Schnorr scheme. Then, based on the result of [25] which shows that under DLP assumption the Schnorr scheme is secure in the random oracle model [4], they concluded that the LKK satisfies all the requirements from R1 to R5. However, the proofs provided in [15] is not rigorous. For example, when they discuss the forgery by Alice in Theorem 1 (page 480 of [15]), the ability of Alice is limited to forge a Schnorr signature pair (r, s) for a new message m such that $g^s = (y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B)^{h(m,r)} \cdot r \bmod p$, where Alice knows two exponents k_A and k such that $r_A = g^{k_A} \bmod p$ and

$r = g^k \bmod p$. However, according to the verification procedure described in their scheme, a successful forgery by Alice only means that she can generate a Schnorr signature pair (r, s) for a new message m that satisfies $g^s = (y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \cdot y_B)^{h(m,r)} \cdot r \bmod p$, where the warrant $\bar{m}_w$ may be different from m_w though m conforms to $\bar{m}_w$, and $\bar{r}_A$ and r are two public values but Alice maybe *does not* know their discrete logarithms to the base g (In the extreme situation, $\bar{r}_A, r \notin G_q$). In fact, the original signer Alice can mount the following concrete attack.

(1) Create a warrant $\bar{m}_w$, select a random number $c \in \mathbb{Z}_q^*$, and then define $\bar{r}_A = y_B^{-1} \cdot g^c \bmod p$.
(2) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is given by:

$$\bar{x}_P = c + x_A \cdot h(\bar{m}_w, \bar{r}_A) \bmod q, \quad \bar{y}_P = g^{\bar{x}_P} \bmod p.$$

One can directly verify that $(\bar{x}_P, \bar{y}_P)$ is a valid proxy key pair with respect to $(\bar{m}_w, \bar{r}_A)$, since $\bar{y}_P = g^{\bar{x}_P} \bmod p = y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \cdot y_B \bmod p$.

After getting $\bar{x}_P$, for any message m which conforms to $\bar{m}_w$, Alice can generate a valid proxy signature σ for m in which Bob will be designated as the proxy signer. Bob is thus framed that he signed some message on behalf of Alice. When such a forged proxy signature is presented, a verifier cannot recognize that in fact it was not generated by Bob. In the above attack, Alice cannot provide $\bar{s}_A$ satisfying $g^{\bar{s}_A} = y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \bmod p$. So, this attack is more powerful if the original signer is not supposed to present s_A when a dispute between Alice and Bob happens. For example, in electronic commerce setting where mobile agents are used to transfer signed warrants [15], all or some s_A's could be deleted before potential disputes occur due to the storage consideration. Anyway, neither the original LKK scheme nor the revised LKK scheme presented how to solve disputes between the original signer and proxy signers. At the same time, the authors also did not mention whether there is a need to keep s_A and x_P after they have been used.

Lee et al. [16] found that in the Schnorr-based LKK scheme, the original signer can change Bob's standard Schnorr signatures into proxy signatures in which Bob is designated as the proxy signer, and vice versa. We call this attack *Transferring Attack*. Their attack is weaker than our above attack in two respects. On the one hand, our attack allows the original signer to forge proxy signature on *any* message as her will, but in the transferring attack she cannot forge proxy signatures for all messages which the proxy signer never signs. On the other hand, we note that the transferring attack can be eliminated by a simple technique - padding. For example, we can modify the LKK scheme as follows. When the proxy signer Bob wants to generate a proxy signature for a message m, he signs on the padded message $m||m_w$, instead of m. That is, here, the warrant m_w is viewed as a proxy signature identifier. At the same time, a proxy signer should not generate standard signatures on a message of the form $m||m_w$. With this modification, it is now difficult for the original signer Alice to apply transferring attacks. In addition, Sun et al. also find a simple attack the Schnorr-based LKK scheme [30].

Furthermore, to remove the secure channel and avoid the transferring attack of the original LKK scheme, Lee et al. proposed a revised LKK scheme in [16]. In the next subsection, however, we will show that their revised LKK scheme is still insecure.

3.2 Revised LKK Scheme and Its Security

Lee et al.'s revised LKK scheme [16] is reviewed as follows.

Revised Proxy Key Generation: Bob gets a proxy key pair (x_P, y_P) from the original signer Alice as follows.

1) Alice chooses at random $k_A \in \mathbb{Z}_q^*$, and computes $r_A = g^{k_A} \bmod p$ and $s_A = k_A + x_A \cdot h(m_w, r_A) \bmod q$. Then, the tuple (m_w, r_A, s_A) is sent to Bob through a public channel.
2) Bob checks whether (r_A, s_A) is a valid Schnorr signature on the warrant m_w. If the answer is positive, Bob chooses a random number k_P, computes $r_P = g^{k_P} \bmod p$, and then sets his proxy key pair (x_P, y_P) as

$$x_P = s_A + r_P \cdot x_B \bmod q, \quad \text{and} \quad y_P = g^{x_P} \bmod p.$$

Proxy Signature Generation and Verification: Using the proxy private key x_P and public parameters (m_w, r_A, r_P), the proxy singer Bob can generate a standard DLP-based signature for any message m which conforms to the warrant m_w. Let $\sigma = (sing(m, x_P), m_w, r_A, r_P)$ [1] be Bob's proxy signature on message m, a verifier checks the validity of σ with respect to the proxy public key y_P, which is computed by:

$$y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B^{r_P} \bmod p \ (= g^{x_P} \bmod p). \tag{6}$$

Lee et al. claimed that the above revised LKK scheme not only removes the requirement on the secure channel, but also overcomes the weaknesses in previous schemes. They indeed provide a security analysis to show that the revised LKK scheme is immune to the transferring attack. However, we find their revised version is still vulnerable to the forgery attack by the original signer and the original signer changing attack.

Forgery by the Original Singer. To forge a proxy key pair, the original signer Alice selects two random numbers $\bar{k}_P, c \in \mathbb{Z}_q^*$, sets $\bar{r}_P = g^{\bar{k}_P} \bmod p$ and defines $\bar{r}_A = y_B^{-\bar{r}_P} \cdot g^c \bmod p$. Then, Alice computes the forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ as

$$\bar{x}_P = c + x_A \cdot h(\bar{m}_w, \bar{r}_A) \bmod q, \quad \bar{y}_P = g^{\bar{x}_P} \bmod p.$$

In the above equation, $\bar{m}_w$ is a new warrant created by Alice. In addition, the forged $(\bar{x}_P, \bar{y}_P)$ with $(\bar{m}_w, \bar{r}_A, \bar{r}_P)$ is a valid proxy key pair, since

$$\begin{aligned} y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \cdot y_B^{\bar{r}_P} &= y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot y_B^{-\bar{r}_P} \cdot g^c \cdot y_B^{\bar{r}_P} \bmod p \\ &= y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot g^c \bmod p \\ &= g^{c + x_A \cdot h(\bar{m}_w, \bar{r}_A)} \bmod p \\ &= g^{\bar{x}_P} \bmod p \\ &= \bar{y}_P. \end{aligned}$$

[1] If y_A and y_B are not included in m_w, it is necessary to list them in σ explicitly.

Using the forged key pair $(\bar{x}_P, \bar{y}_P)$, Alice can create valid proxy signatures for arbitrary messages at her will.

Original Signer Changing Attack. Now, assume that Bob has a proxy key pair (x_P, y_P) for the original signer Alice. At the same time, Bob generated a proxy signature $\sigma = (r, s, m_w, r_A, r_P)$ for a message m using the Schnorr signature scheme. That is, there exists a number $k \in \mathbb{Z}_q^*$ such that

$$r = g^k \bmod p, \quad s = k + x_P \cdot h(m, r) \bmod q.$$

In the following, we show that a third party Charlie can transform σ into $\bar{\sigma}$ such that $\bar{\sigma}$ is a proxy signature generated by Bob on the behalf of Charlie himself. To this end, Charlie first creates a new warrant $\bar{m}_w$ such that the message m conforms to $\bar{m}_w$. In addition, if necessary, Charlie specifies in $\bar{m}_w$ that himself is the original signer, and Bob is the proxy signer. After $\bar{m}_w$ has been prepared, Charlie chooses a random number $k_C \in \mathbb{Z}_q^*$, and computes $r_C = g^{k_C} \bmod p$ and $s_C = k_C + x_C \cdot h(\bar{m}_w, r_C) \bmod q$. Finally, using the public information s_A, Charlie computes $\bar{s}$ by:

$$\bar{s} = s - s_A \cdot h(m, r) + s_C \cdot h(m, r) \bmod q$$

The resulting proxy signature $\bar{\sigma}$ for message m is set as $\bar{\sigma} = (r, \bar{s}, \bar{m}_w, r_C, r_P)$.

To verify the validity of $\bar{\sigma}$, a recipient first generates the proxy public key $\bar{y}_P$ from

$$\bar{y}_P = y_C^{h(\bar{m}_w, r_C)} \cdot r_C \cdot y_B^{r_P} \bmod p.$$

Let $\bar{x}_P = r_P \cdot x_B + s_C \bmod q$, then we have $\bar{y}_P = g^{\bar{x}_P} \bmod p$. The following equalities justify that $\bar{\sigma}$ is a valid proxy signature generated by Bob on behalf of Charlie, i.e., the tuple $(r, \bar{s})$ is a valid Schnorr signature pair with respect to the key pair $(\bar{x}_P, \bar{y}_P)$:

$$\begin{aligned}
\bar{s} &= s - s_A \cdot h(m, r) + s_C \cdot h(m, r) \bmod q \\
&= k + x_P \cdot h(m, r) - s_A \cdot h(m, r) + s_C \cdot h(m, r) \bmod q \\
&= k + (r_P \cdot x_B + s_A) \cdot h(m, r) - s_A \cdot h(m, r) + s_C \cdot h(m, r) \bmod q \\
&= k + (r_P \cdot x_B + s_C) \cdot h(m, r) \bmod q \\
&= k + \bar{x}_P \cdot h(m, r) \bmod q.
\end{aligned}$$

Furthermore, it is not difficult to see that when Bob has two signed warrants (m_w, r_A, s_A) and $(\bar{m}_w, r_C, s_C)$ generated by Alice and Charlie respectively, anybody can convert Bob's proxy signature σ on a message m on behalf of the original signer Alice into a new proxy signature $\bar{\sigma}$ on the same message in which Bob is the proxy signer but Charlie becomes the original signer. The only requirement is that message m also conforms to the warrant $\bar{m}_w$. The reason is that (r_A, s_A) and (r_C, s_C) are in fact public information, since they are transferred through a public channel.

4 The LKK RSA-based Scheme

In this section we analyze the security of the LKK RSA-based proxy scheme proposed in [15]. First of all, we review it briefly.

Let $((n_A, e_A), d_A)$ and $((n_B, e_B), d_B)$ be the certified ordinary RSA key pairs of Alice and Bob, respectively. Denote the identities of Alice and Bob as ID_A and ID_B. To designate Bob as her proxy signer, Alice creates a warrant m_w and signs it, i.e., she generates $k = h(ID_A, m_w)^{d_A} \bmod n_A$. Then, (ID_A, m_w, k) is sent to Bob.

Bob checks the validity of (ID_A, m_w, k) by $h(ID_A, m_w) \equiv k^{e_A} \bmod n_A$. To generate a proxy signature on message m which conforms to m_w, he computes the triple (x, y, z) as follows:

$$\begin{aligned} x &= h(ID_A, m_w, ID_B, m)^{d_B} \bmod n_B, \\ y &= h(ID_A, m_w)^x \bmod n_A, \\ z &= k^x \bmod n_A. \end{aligned}$$

The resulting proxy signature is $\sigma = (ID_A, m_w, ID_B, m, x, y, z)$.

A recipient accepts a proxy signature σ iff all the following checks hold:

1) Verify whether m conforms to the warrant m_w.
2) Verify Bob's RSA signature: $h(ID_A, m_w, ID_B, m) \equiv x^{e_B} \bmod n_B$.
3) Verify the validity of y: $y \equiv h(ID_A, m_w)^x \bmod n_A$.
4) Verify Alice's RSA signature: $y \equiv z^{e_A} \bmod n_A$.

In [15], Lee et al. proved that the above RSA-base proxy signature is as secure as the RSA signature scheme. Obviously, forgery by Alice is difficult since she has to forge Bob's standard RSA signature x on message (ID_A, m_w, ID_B, m). However, whether Bob can mount some attack is another story. Similar to previous discussion presented in Section 3.1, one can find that their proof is not rigorous. And in fact, the reasoning about Bob's attacking ability in the proof of Theorem 2 (page 482 of [15]) is wrong, since given the value of x Bob cannot find the value of $1/x \bmod \phi(n_A)$. In more detail, the authors first concluded that if Bob can generate a valid proxy signature σ without Alice's delegation, then the following equations hold:

$$z = y^{d_A} \bmod n_A = h(ID_A, m_w)^{x d_A} \bmod n_A).$$

Secondly, the authors claimed that the above equalities mean that Bob can compute Alice's RSA signature k on message (ID_A, m_w) from

$$k = z^{1/x} \bmod n_A \; (= h(ID_A, m_w)^{d_A} \bmod n_A.$$

So the authors have assumed that Bob can get the value of $1/x \bmod \phi(n_A)$ when the value of x is given. However, this is the RSA problem [26] so it is an intractable problem for Bob! Therefore, Theorem 2 provided by [15] is invalid.

To show the insecurity of the RSA-based LKK scheme directly, we demonstrate the following attack for Bob who is not designated as a proxy signer of Alice.

(1) Try different warrants to get an m_w such that m conforms to m_w, and that $e_A | x$. The latter condition means that $x = h(ID_A, m_w, ID_B, m)^{d_B} \bmod n_B = e_A \cdot b$ for an integer b.
(2) Compute $y = h(ID_A, m_w)^x \bmod n_A$ and $z = h(ID_A, m_w)^b \bmod n_A$.
(3) Output the proxy signature $\sigma = (ID_A, m_w, ID_B, m, x, y, z)$.

The correctness of the above attack is easy to check, since $y \equiv h(ID_A, m_w)^x \bmod n_A = h(ID_A, m_w)^{e_A \cdot b} \bmod n_A = z^{e_A} \bmod n_A$. The smaller e_A is, the easier is to mount our attack. Suggested values for e_A in practice include 3, 17, and $2^{16} + 1 = 65537$ (X.509 recommends 65537, and PKCS #1 recommends either 3 or 65537). If $e_A = 3$, our attack succeeds for one try of m_w with probability about 1/3. Even for $e_A = 65537$, one required warrant can be found after about $2^{16} + 1$ different tries.

The LKK RSA-based scheme has another weakness, that is, it is not very efficient. In both proxy signature generation and verification, one entity has to perform three RSA modular exponentiations. Using the generic construction given in [5], one can derive a provable secure RSA-based proxy signature scheme in which only two modular exponentiations are needed in both proxy signature generation and verification. The basic idea is very simple. That is, to delegate her signing ability to Bob, Alice sends Bob a proxy certificate `cert` which is her standard RSA-signature on the warrant m_w. A proxy signature σ on a message m consists of $(m, s, m_w, \texttt{cert})$, where s is Bob's standard RSA signature on message m. A verifier validates σ by performing the following three checks: (a) Whether m conforms to the warrant m_w; (b) Whether `cert` is Alice's RSA signature on the warrant m_w; (c) And whether s is Bob's RSA signature on the message m. To provide provable security, m_w and m are padded in some way. For more detail, please refere to [5].

5 Conclusion

In this paper, we presented a security analysis of four proxy signature schemes newly proposed in [15, 16]. Our results show that all these schemes are insecure, i.e., forgeable. As a by-product, the fully distributed proxy signature scheme in [11] is also broken, because it is based on the insecure LKK scheme [15]. In addition, we pointed out that the security proofs provided in [15] are incorrect.

References

1. C. Adams and S. Farrell. Internet X.509 public key infrastructure: Certificate management protocols. RFC 2510, March 1999.
2. M. Ai-Ibrahim and A. Cerny. Proxy and threshold one-time signatures. In: *Proc. of the 1th International Conference Applied Cryptography and Network Security* (*ACNS'03*), LNCS 2846, pp. ??-?? (to appear). Springer-Verlag, 2003.
3. A. Bakker, M. Steen, and A.S. Tanenbaum. A law-abiding peer-to-peer network for free-software distribution. In: *IEEE International Symposium on Network Computing and Applications* (*NCA'01*), pp. 60-67. IEEE, 2001.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In: *Proc. of 1st ACM Conference on Computer and Communications Security* (*CCS'93*), pp. 62-73. ACM Press, 1993.
5. A. Boldyreva, A. Palacio, and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. Available at `http://eprint.iacr.org/2003/096`
6. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, July 1985, IT-31(4): 469-472.
7. FIPS 186. *Digital Signature Standard*. U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, 1994.

8. I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In: *Proc. 5th ACM Conference on Computer and Communications Security* (*CCS'98*), pp. 83-92. ACM Press, 1998.
9. H. Ghodosi and J. Pieprzyk. Repudiation of cheating and non-repudiation of Zhang's proxy signature schemes. In: *Information Security and Privacy* (*ACISP'99*), LNCS 1587, pp. 129-134. Springer-Verlag, 1999.
10. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, April 1988, 17(2): 281-308.
11. J. Herranz and Sáez. Verifiable secret sharing for general access structures, with applications to fully distributed distributed proxy signatures. In: *Financial Cryptography* (*FC'03*), LNCS 2742, pp. 286-302. Springer-Verlag, 2003.
12. J. Herranz. Personal communications. August, 2003.
13. S. Kim, S. Park, and D. Won. Proxy signatures, revisited. In: *Information and Communications Security* (*ICICS'97*), LNCS 1334, pp. 223-232. Springer-Verlag, 1997.
14. B. Lee, H. Kim, and K. Kim. Strong proxy signature and its applications. In: *Proceedings of the 2001 Symposium on Cryptography and Information Security* (*SCIS'01*), Vol. 2/2, pp. 603-608. Oiso, Japan, Jan. 23-26, 2001.
15. B. Lee, H. Kim, and K. Kim. Secure mobile agent using strong non-designated proxy signature. In: *Information Security and Privacy* (*ACISP'01*), LNCS 2119, pp. 474-486. Springer-Verlag, 2001.
16. 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, pp. 68-79. Springer-Verlag, 2003.
17. N.-Y. Lee, T. Hwang, and C.-H. Wang. On Zhang's nonrepudiable proxy signature schemes. In: *Information Security and Privacy* (*ACISP'98*), LNCS 1438, pp. 415-422. Springer-Verlag, 1998.
18. J. Leiwo, C. Hanle, P. Homburg, and A.S. Tanenbaum. Disallowing unauthorized state changes of distributed shared objects. In: *Information Security for Global Information Infrastructures (SEC'00)*, pp. 381-390. Kluwer, 2000.
19. M. Mambo, K. Usuda, and E. Okamoto proxy signature: Delegation of the power to sign messages. *IEICE Trans. Fundamentals*, Sep. 1996, Vol. E79-A, No. 9, pp. 1338-1353.
20. M. Mambo, K. Usuda, E. Okamoto. Proxy signatures for delegating signing operation. In: *Proc. of 3rd ACM Conference on Computer and Communications Security* (*CCS'96*), pp. 48-57. ACM Press, 1996.
21. M. Meyers, C. Adams, D. Solo, and D. Kemp. Internet X.509 certificate request format. RFC 2511, March 1999.
22. T. Okamoto, M. Tada, and E. Okamoto. Extended proxy signatures for smart cards. In: *Information Security Workshop* (*ISW'99*), LNCS 1729, pp. 247-258. Springer-Verlag, 1999.
23. 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, pp. 451-455. Springer-Verlag, 2001.
24. H. Petersen and P. Horster. Self-certified keys - concepts and applications. In: *Proc. of 3rd Conference on Communications and Multimedia Security*, pp. 102-116. Chapman & Hall, 1997.
25. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3): 361-369, 2000.
26. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, Feb. 1978, 21(2): 120-126.
27. Z. Shao. Proxy signature schemes based on factoring. *Information Processing Letters*, 2003, 85: 137-143.
28. C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptography*, 1991, 4(3): 161-174.
29. H.-M. Sun. Threshold proxy signatures. *IEE Proc.-Computers & Digital Techniques*, Sept. 1999, 146(5): 259-263.
30. H.-M. Sun and B.-T. Hsieh. On the security of some proxy signature schemes. Available at `http://eprint.iacr.org/2003/068`.
31. H. Wang and J. Pieprzyk. Efficient one-time proxy signatures. In: *Asiacrypt'03* (to appear). Springer-Verlag, 2003.
32. K. Zhang. Threshold proxy signature schemes. In: *Information Security Workshop* (*ISW'97*), LNCS 1396, pp. 282-290. Springer-Verlag, 1997.

33. K. Zhang. Nonrepudiable proxy signature schemes. Manuscript, 1997. Available at http://citeseer.nj.nec.com/360090.html