A New Forward Secure Proxy Signature Scheme

Ming-Yang Chen*, Wen-Chung Kuo**, Chi-Sung Laih*

*Department of Electrical Engineering,
National Cheng-Kung University, Taiwan, R.O.C.
E-mail: cmy@crypto.ee.ncku.edu.tw; laihcs@eembox.ncku.edu.tw

**Department of Computer Science and Information Engineering,
National Formosa University, Taiwan, R.O.C.
E-mail: simonkuo@sunws.nfu.edu.tw
A New Forward Secure Proxy Signature Scheme

Abstract
Since the concept of the proxy signature scheme was proposed by Mambo et al. in 1996, it is widely used for people to delegate their own power to others. So far, the proxy signature scheme is only applied in special time period, such as when the original signer is not in his office or when he travels outside. Then, the original signer will use the same secret proxy key in the next special time period. However, those proxy schemes appear to be secure today, they may be exploded in the future while the secret key is exposed. As far as the author is aware, this type of proxy signature schemes with forward-secure ability has not appeared in the literature. Therefore, a proxy signature scheme with the forward-secure ability is proposed in this article. It provides not only the original signer secure keys with forward-secure ability but also the proxy signers secure keys with the forward-secure ability.

Keywords: proxy signature scheme, forward-secure.

1 Introduction
Delegating the authority of signature to a proxy is useful in many scenarios. For example, a manager can delegate his secretaries or his subordinates to sign certain documents when he travels outside. Obviously, there are many technical challenges when the original signer delegates the authority of signing documents to proxy signers. In 1996, Mambo, Usudo and Okamoto [9, 10] gave a systematic discussion delegation models and then firstly introduced the concept of proxy signature. There are three levels of delegation in MUO-scheme [9]:

1)Full delegation: The original signer gives its own private key to the proxy signer and then the proxy signer can use this private key to create the same signature as the one that the original signer creates.
2)Partial delegation: The original signer generates a proxy signature key by using its private key and gives it to the proxy signer. The proxy signer then uses the proxy key to sign the document. Verifier can distinguish who created this signature.
3)Delegation by warrant: Warrant is a certificate composed of a message part and a public signature key. The proxy signer obtains the warrant from the original signer and uses the corresponding private key to sign the documents. The resulting signature must include both the created signature and the warrant.

Since MUO-scheme was announced, many proxy signature schemes have been proposed. In [7], Kim et al. (KPW-scheme) used the Schnorr signature and the warrant to propose a
proxy signature scheme in partial delegation with warrant. However, in 1999, Sun and Hsieh [15] pointed out that the MUO-scheme [9] is unfair to the original signer because the proxy signer can transfer the signing right to others, and that the KPW-scheme is insecure to the public key substitution attack in which an attacker can create a valid proxy signature by updating his own public key with other’s public key. To deal with these security flaws, Sun and Hsieh also proposed two modified schemes [15, 16]. Recently, Lee et al. [8] defined the following properties that a strong proxy signature scheme should provide.

1) **Strong unforgeability**: Only legitimate proxy signers can create the valid signature. Even the original signer can not forge the proxy signer to generate the valid signature. At the same time, the original signer or third parties who are not designated as a proxy signer cannot create a valid proxy signature.

2) **Verifiability**: Everyone who can verify the signature and the signed message should conform to the delegation warrant.

3) **Strong identifiability**: Everyone can authenticate the identity of the corresponding proxy signer from a proxy signature. In other words, everyone can determine who signs the message in what period of time.

4) **Strong undeniability**: The proxy signer can not repudiate the signature which he once generated.

5) **Prevention of misuse**: The proxy key pair should be used in any place which conforms to the warrant.

In order to solve the delegation problem, the proxy signature scheme has been proposed and the proxy scheme is only applied in the special time period, such as when the original signer is not in his office or he travels outside. The original signer needs to revoke the delegation right from proxy signers when he is on duty. Similarly, the original signer will give the delegation right to proxy signers when he is absent again. Currently, the proxy protocols can be classified into two approaches. One approach is to change the public key of the original signer. This approach is impractical because we can not promise that the signature signed later is valid when the signer’s secret key is exposed. In addition, all signature generated earlier by the original signer cannot be verified when the signer’s public key is changed. The other approach is the proxy scheme with the forward-secure ability, that is, the proxy signers can sign the message with different secret proxy key in any period of time. In this way, although the current proxy signing key is exposed, the verifier can still verify the signature signing at next time period.

As to the best of our knowledge, the proxy signature scheme with forward-secure ability has not yet been proposed. Due to this reason, we will add the forward-secure property to the proxy signature scheme in this article. The idea of forward-secure signature scheme is firstly proposed by Anderson in [3] and formalized by Bellare and Miner in [4]. Namely, the goal of
the forward-secure signature scheme is to preserve the validity of past signatures even if the current secret key has been compromised. In the forward-secure signature scheme, there are many time periods, and each time period requires a unique secret key, but the public key remains fixed for different secret keys. Each subsequent secret key is computed from the current secret key via a famous key update algorithm [1, 2].

The remainders of this paper are organized as follows. First, we will give some related works in Section 2. Then, we present our scheme in Section 3, and we analyze the security of our scheme in Section 4. Finally, our conclusion is illustrated in Section 5.

2 Review of the forward-secure signature scheme

In 1997, Anderson [3] proposed a new digital signature scheme with forward secure ability which refers to the situation that even the attacker who breaks the current secret key cannot forge the past signature which is signed by the secret key of prior intervals. In order to have this forward-secure ability, the user’s secret key must be changed periodically when the time interval changes by using a one way hash function \( h \). From Figure 1, the secret key \( SK_t \) in the period \( t \) is calculated from the period \( t-1 \), i.e. \( SK_t = h(SK_{t-1}) \). Now, if the secret key \( SK_t \) of period \( t \) leaks out or is broken by an attacker, we must make sure that the attacker cannot forge the signatures corresponding to related secret keys.

![Figure 1 Forward Secure signature scheme](image)
Here, the flow chart of the original signer will update his own secret key when the period of time changed is shown in Fig.2. Then, the proxy signer’s proxy key would be changed if the secret key of the original signer changed. This method is just a general and well-known forward-secure signature scheme. Now, we will extend this ideal to proxy signer, i.e., not only the original signer has the forward ability but also the proxy signers have the same right. From Fig.3, there are $N$ proxy signers who are delegated by the original signer to sign documents in different time period $T$. This forms a two-dimensional architecture instead of just one-dimensional like Fig.2.

3 Proposed a forward-secure proxy signature scheme

3.1 System parameters Definitions

There are some parameters used in our paper unless otherwise specified:

- The suffix “o” represents the original signer. For instance, $SK_{o,1}$ and $t_{o,1}$ represent the original signer’s secret key and his time seeds, respectively.
- The suffix “i” stands for proxy signers. For example, $SK_{i,1}$ and $t_{i,1}$ stands for proxy signer’s secret key and his own time seeds, respectively.

3.2 Our proposed scheme

Here, we assume that there are one original signer, one absolute secure KDC, which is the most important role since it is trusted by everyone, and $N$ proxy signers in our scheme. In other words, we assume that the attacker can not break the KDC and get nothing from KDC.

In order to illustrate this clearly, we divide our scheme into four phases as follows. Phase1 is the public key generation and Phase2 the original signer and the proxy signer key distribution.
and their key update. In Phase3, the original signer generates the delegation signature and sends it to the proxy signer and then the proxy signer verifies this delegation signature. In Phase4, the proxy signer generates the proxy signature and sends it to verifier. Then every verifier can prove the proxy signature is valid or not.

**Phase1: Public Key Generation.** Public key generating algorithm generates a common public key for each proxy signer, and also creates secret key seeds for each proxy signer to update his own secret key while the time period has changed. KDC will distribute those secret key seeds to all proxy signers. This algorithm will also generate a lot of key seeds and two public keys, one is for proxy signers, and the other is for the original signer. The steps of this algorithm are shown as follows:

**Step 1.** KDC randomly selects two primes \( p \) and \( q \), then computes \( n = pq \).

**Step 2.** KDC selects a number \( g_0 \) such that \( g_0 \in \mathbb{Z}_n^* \) and uses the seed \( \beta_o \) to generate \( T_0 \) primes \( e_1, e_2, \cdots, e_{T_0} \), where \( e_i \geq 2^l \) and \( \gcd(e_i, \phi(n)) = 1 \) for \( 1 \leq i \leq T_0 \).

**Step 3.** KDC let the secret key \( t_{0,1} = g_0 \mod n \) for the original signer.

**Step 4.** Computes the public key for the original signer, \( u = 1 / \prod_{i=1}^{T_0} g_0^{e_i} \mod n \) such that 
\[
 s_{0,i}^e u = 1.
\]

**Step 5.** Randomly selects primes \( b_{1,1}, b_{2,1}, \cdots, b_{1,j}, \cdots, b_{N,T} \), where \( b_{i,j} \geq 2^l \) and \( \gcd(b_{i,j}, \phi(n)) = 1 \) for \( 1 \leq i \leq N, 1 \leq j \leq T \).

**Step 6.** Calculates \( t_{i,1} = \prod_{j=1}^{T_0} b_{i,j} \mod n \) and public key \( v \) for all proxy signers, \( v = 1 / \prod_{i=1}^{T_0} \prod_{j=1}^{T_0} b_{i,j} \mod n \) such that \( t_{i,j} v = 1 \). Then KDC publishes all \( b_{1,1}, b_{2,1}, \cdots, b_{i,j}, \cdots, b_{N,T} \) data.

**Step 7.** KDC sends \((t_{0,1}, \beta_0)\) to the original signer through secure channels.

KDC also sends \( t_{i,1}, t_{2,1}, \cdots, t_{N,1} \) to each proxy signer \( P_i \), where \( 1 \leq i \leq N \).

**Phase2: the original signer and the proxy signer key distribution and their keys update.**
In this phase, the original signer generates his own secret key in the first period of time, and each proxy signer generates his own secret key corresponding to first period, which is different from the original signer’s period.

**For the original signer:**

**Step 1.** The original signer computes the secret key, \( s_{0,1} = t_{0,1}^{e_{0,1}} \mod n \), for the first period.

**Step 2.** If the original signer wants to generate the secret key seed by using \( t_{0,2} = t_{0,1}^{e_{0,2}} \mod n \) for next period, the secret key would be \( s_{0,2} = t_{0,2}^{e_{0,2}} \mod n \) in the second period, etc. So, the original signer can update his own secret key from period \( l \) to \( l+1 \) by computing: \( s_{0,l+1} = t_{0,l+1}^{e_{0,l+1}} \mod n \) and \( t_{0,l+2} = t_{0,l+1}^{e_{0,l+1}} \mod n \).

**For the proxy signer \( P_i \):**

**Step 1.** \( P_i \) generates his secret key, \( s_{i,1} = t_{i,1}^{e_{i,1}} \mod n \), for the first period.

**Step 2.** If the proxy signer \( P_i \) wants to generate his secret key seed by computing \( t_{i,2} = t_{i,1}^{e_{i,2}} \mod n \) for the next period and then the secret key in the second period would be \( s_{i,2} = t_{i,2}^{e_{i,2}} \mod n \). So, the proxy signer \( P_j \) can update his own secret key from period \( j \) to \( j+1 \) by using: \( s_{i,j+1} = t_{i,j+1}^{e_{i,j+1}} \mod n \) and \( t_{i,j+2} = t_{i,j+1}^{e_{i,j+1}} \mod n \).

**Phase 3a: Generate the original signer’s warrant.**

The original signer generates the signature corresponding to each proxy signer who has the capability to sign the signature on the behalf of the original signer.

**Step 1.** The original signer randomly selects \( r_{a,i} \), where \( r_{a,j} \in \mathbb{Z}_n^* \) for \( 1 \leq i \leq N \).

**Step 2.** Calculates \( y_{o,i} = r_{o,i} \mod n \) and \( \sigma_{o,i} = H(i, e_i, y_{o,i}) \), then he can compute \( z_{o,i} = r_{o,i}^{\sigma_{o,i}} \mod n \).

Hence, the delegation signature corresponding to \( P_i \) is \( \zeta_{o,i} = (i, z_{o,i}, \sigma_{o,i}, e_i) \).
Phase 3b: Verify original signer’s signature.

Each proxy signer receives the signature $\zeta_{o,i}$ from the original signer, and then he can check whether the signature $\zeta_{o,i}$ is valid or not by the following steps.

**Step 1.** Proxy signer $P_i$ computes $y'_{o,i} = z_{o,i}^a u^{\sigma_{o,i}} \mod n$ after receiving $\zeta_{o,i}$ from the original signer.

**Step 2.** He checks whether $\sigma_{o,i} = H(i, e_j, y'_{o,i})$ is true or not.

**Step 3.** If the signature is valid then $P_i$ accepts $\zeta_{o,i}$, otherwise rejects this proxy signature.

So, if the signature is valid, each proxy signer will keep $\zeta_{o,i} = (i, z_{o,i}, \sigma_{o,i}, e_i)$ in mind, and when he generates the proxy signature, he will utilize it.

Phase 4a: Generate the proxy signature from the proxy signer.

Each proxy signer uses his own secret key and the delegation signature $\zeta_{o,i} = (i, z_{o,i}, \sigma_{o,i}, e_i)$ from the original signer to sign the message $M$.

**Step 1.** $P_i$ uses $(b_{i,j}, s_{i,j})$ to sign $M$.

**Step 2.** $P_i$ also selects random numbers $r_i \in Z_n^*$, then he computes $y_j = r_j^{h_{o,j}} \mod n$,

$$z_j = r_i s_{i,j}^{e_i} \mod n \quad \text{and} \quad \sigma_j = H(i, j, b_{i,j}, y_j, z_{o,i}, y_{o,j}, M).$$

**Step 3.** The original signer delegates the proxy signature of message $M$ corresponding to $P_i$, $\zeta_i = (i, j, b_{i,j}, z_i, \sigma_i, e_i, z_{o,i}, \sigma_{o,i})$ to the proxy signers.

Phase 4b: Verification of proxy signature $\zeta_i$.

The verifier should verify not only the validation of the signature, but also the warrant corresponding to each proxy signer.

**Step 1.** The verifier computes $y'_{o,i} = z_{o,i}^a u^{\sigma_{o,i}} \mod n$ after receiving $\zeta_i$ from the proxy
signer. Then he checks whether \( H(i, e_i, y'_{o,i}) = \sigma_{o,i} \) or not.

**Step 2.** Calculates \( y'_i = z_i^{h_i} v^{e_i} \mod n \).

**Step 3.** Computes \( H(i, j, b_{i,j}, y'_{i}, z_{o,i}, y'_{o,i}, M) \).

Therefore, if \( \sigma_{i} \) is equal to \( H(i, j, b_{i,j}, y'_{i}, z_{o,i}, y'_{o,i}, M) \), then each verifier can prove that the proxy signature is delegated by the original signer. Otherwise, the verifier will give up this proxy signature from the proxy signer.

### 4 System analysis

Until now, we have proposed a proxy signature scheme with forward-secure ability. The major feature of our scheme is that multi-signers can sign the message on the behalf of the original signer. However, we can not promise that all the proxy signers are honest. Therefore, our scheme might seem to be vulnerable to proxy signers’ collusion attacks. Here, we will make sure that even though there are \( N-1 \) proxy signers colluded, those proxy signers can not discover the legal signer’s secret key. In Section 4.1, we give some attacks to our proposed scheme and inspect whether it can prevent these attacks or not. Then, we will check whether our scheme satisfies the requirements of proxy signature scheme [8] or not in Section 4.2.

#### 4.1 Cryptanalysis

In order to explain how the collusion attacks might be launched in our scheme, we assume that there are an original signer and three proxy signers in our scheme. Therefore, the original signer has three secret keys, \( (e_1, e_2, e_3) \) and the three proxy signers (such as \( P_1, P_2, \) and \( P_3 \)) also have three different information \( (b_{1,1}, b_{1,2}, b_{1,3}), (b_{2,1}, b_{2,2}, b_{2,3}) \) and \( (b_{3,1}, b_{3,2}, b_{3,3}) \) in different period, respectively.

**Attack 1:** Two proxy signers, \( P_1 \) and \( P_2 \), want to derive \( P_3 \)’s secret key \( s_{3,1} \) from their corresponding secret keys \( s_{1,1} \) and \( s_{2,1} \).

In fact, \( s_{1,1} \) and \( s_{2,1} \) represent the secret keys of \( P_1 \) and \( P_2 \), where \( s_{1,1} = g^{b_{1,1} h_1 h_2 h_3, h_1 h_2 h_3, h_1 h_2 h_3} \mod n \) and \( s_{2,1} = g^{b_{2,1} h_1 h_2 h_3, h_1 h_2 h_3, h_1 h_2 h_3} \mod n \), and their security is based on factoring problem. Obviously, \( P_1 \) and \( P_2 \) want to derive \( P_3 \)’s secret key \( s_{3,1} \) from their secret keys \( s_{1,1} \) and \( s_{2,1} \), and in order to find out the secret key \( s_{3,1} \), they will also face the problem of factoring \( n \). Even if there are \( N-1 \) proxy signers colluding, they still can not having the access to derive the \( N \)th legal proxy signer’s secret key.
**Attack 2:** Two proxy signers, $P_1$ and $P_2$, want to find some useful information from their corresponding secret keys $s_{1,3}$ and $s_{2,3}$.

In this attack, the proxy signers $P_1$ and $P_2$ use the extended Euclidean algorithm to find the $\text{gcd}(s_{1,3}, s_{2,3}) = g^{h_1 h_2 h_3 h_4 h_5 h_6} \mod n$ from their secret key $s_{1,3}$ and $s_{2,3}$, where $s_{1,3} = g^{h_1 h_2 h_3 h_4 h_5 h_6} \mod n$ and $s_{2,3} = g^{h_1 h_2 h_3 h_4 h_5 h_6} \mod n$. However, the proxy signers $P_1$ and $P_2$ do not get anything from the value of $\text{gcd}(s_{1,3}, s_{2,3})$ since it is still based on RSA assumption.

**Attack 3:** A cryptanalyst wishes to find out the past secret information of the original signer from the present period secret information.

Now, we assume that the cryptanalyst can guess the original signer’s secret information $s_{o,2}$ in the period 2. However, he can not carry out the secret keys $s_{o,1} = g^{s_{o,1}} \mod n$ from the secret information $s_{o,2} = g^{s_{o,2}} \mod n$, because the proposed security of this scheme is based on the factoring problem.

### 4.2 Satisfying the Requirements of Proxy Signature

A strong proxy signature scheme must have some of the properties proposed by Lee et al. [8]. Here, we will provide a detailed security discussion on the proposed scheme and claim that our scheme meets all the security requirements listed in [8].

(i) **Strong Unforgeability:** The original signer and the proxy signers have different RSA bases, $g_o$ and $g$, in the proposed scheme. So, the original signer can not get the secret key of the proxy signers unless he steals their secret key from the proxy signers. Therefore, the original signer can not generate a valid proxy signature and claims that it is signed by proxy signers.

(ii) **Verifiability:** Verifiability which means that the signature can be verified by anyone and can check the rights of the proxy signers. Therefore, anyone has the proxy signer’s public key $v$ can verify the signature signed by whom and whether he has the rights to sign the document or not in our scheme. Thus the original signer cannot deny his agreement.

(iii) **Strong Identifiability:** The original signature is distinguishable from the proxy signature because the warrant and the public keys of the original signers and the proxy signers must occur in the proxy signature. In other word, the verifier can verify that whether the proxy signer has the capability to sign the message by computing...
\( \sigma_{a,i} = H(i, e_i, y_{a,i}) = \sigma_{a,i}, \) and identify who signs in what period by the parameter \( b_{i,j}. \)

(iv) **Strong Undeniability**: Once the proxy signer creates a valid proxy signature, he cannot repudiate it because the verifier can check whether \( \zeta_i \) is valid or not when he receives the signature form the proxy signers. Obviously, the verifier can make sure that the signature is signed by user \( i \) in \( j \) period by using the steps of Phase 4b.

(v) **Prevention of Misuse**: In this paper, the secret keys are directly distributed from KDC to the original signer and the proxy signers. In this way, the proxy signer cannot sign certain messages that have not yet been authorized by the original signer.

5 Conclusion

We have proposed a new proxy signature scheme with forward secure ability in this paper. Then, we cryptanalyze the security of our schemes by using some attacks. In addition, our scheme meets all the security requirements such as **strong unforgeability**, **verifiability**, **strong identifiability**, **strong undeniability** and **prevention of misuse** proposed by Lee et al.

**Acknowledge**: This research is supported by National Science Council, Taiwan, R.O.C., under contract no.: NSC93-2218-E-150-013.

References


