

Proxy Signature Schemes for Controlled Delegation

N.R.Sunitha

Department of Computer Science & Engg.
Siddaganga Institute of Technology,
Tumkur, Karnataka, India.

B.B.Amberker

Department of Computer Science & Engg.
National Institute of Technology,
Warangal, Andhra Pradesh, India.

Abstract: A proxy signature scheme allows one user to delegate his/her signing capability to another user called a proxy signer in such a way that the latter can sign messages on behalf of the former. After verification the verifier is convinced of the original signer's agreement on the signed message. We begin by proposing a simple proxy signature scheme which satisfies the basic requirements of a secure proxy signature scheme. We find that the scheme can be used to control delegation of financial power to a proxy signer. Using our scheme the proxy signer will be able to submit an e-cheque for only the amount he is entitled to by the original signer. Any cheating by the original signer or the proxy signer is identified by the verifier i.e. the bank. Any discrete log based signature scheme can be used to sign the messages. Here we use the Digital Signature Algorithm(DSA).

Forward-Secure signatures enable the signer to guarantee the security of messages signed in the past even if his secret key is exposed today. We extended our work on proxy signatures by constructing two forward secure proxy signature schemes, one based on DSA and the other based on the popular Bellare-Miner Forward-secure scheme. Compared to existing schemes, the special feature of our schemes is that an original signer can delegate his signing capability to any number of proxy signers in varying time periods. Though the original signer gives proxy information to all the proxy signers at the beginning of the protocol, the proxy signers will be able to generate proxy signatures only in their allotted time periods. Moreover, our schemes meet the basic requirements of a proxy signature scheme along with proxy revocation. Both on-demand proxy revocation i.e. whenever the original signer wants to revoke the proxy signer and automatic proxy revocation i.e. immediate revocation after the expiry of the time period of the proxy signer, is provided. Additional properties of our scheme are as follows: identity of the proxy signer is available in the information sent by original signer to proxy signer, original signer need not send the information to proxy signer through a secure channel, warrant on the delegated messages can be specified, original signer cannot play the role of proxy signer, and verifier can determine when the proxy signature was generated. The Bellare-Miner Forward-secure proxy scheme can also be used as a Forward-secure threshold proxy signature scheme. We have considered Forgery by the original signer, Impersonating and framing attack to prove the security of our scheme.

Keywords : Proxy Signature, Forward-Security, Proxy revocation, Threshold proxy signature scheme, Digital Signature.

I. Introduction

A proxy signature [16, 17] allows one user Alice, called the original signer, to delegate her signing capability to another user Bob, called the 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. By this the verifier is convinced of the original signer's agreement on the signed message. Proxy signatures can be used in a number of applications like e-cash, electronic commerce, distributed shared object systems etc.

The basic working of most 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 and then checks its validity according to the corresponding standard signature verification procedure.

Mambo, Usuda and Okamoto introduced the concept of proxy signatures and proposed several constructions in [16]. Based on the delegation type, they classified proxy signatures as full delegation, partial delegation and delegation by warrant schemes. In full delegation, Alice's private key is given to Bob so that Bob has the same signing capability as Alice. But 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 can sign any message of his choice i.e. there is no limit on the range of messages he can sign. This limitation is eliminated in delegation by warrant schemes by adding a warrant that specifies what kind of messages are delegated and may contain the identities of Alice and Bob, the delegation period, etc.

Followed by the first constructions given in [16, 17], a number of new schemes and improvements have been proposed [5, 7, 10, 11, 14, 15, 18–20, 22, 24]; however, most of them do not fully meet all the security requirements of a proxy signature scheme (see Section 2).

In [11], Kim, Park and Won proposed a threshold proxy signature, in which the original signing power is shared among a delegated group of n proxy signers such that only t or more of them can generate proxy signatures cooperatively. In [15], 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 [5]. One-time proxy signatures are studied in [8,22]. In [10], 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 [16] and the LKK scheme [5, 15] all are insecure. To remove the requirement of a secure channel and overcome some other weaknesses, they revised the MUO and LKK schemes. In contrast to the above mentioned schemes, which all are based on discrete logarithm cryptosystems, several RSA-based proxy signature schemes are proposed in [5, 19]. The first work to formally define the model of proxy signatures is the recent work of Boldyreva, Palacio, and Warinschi [4]. The authors of [21] provide the first definition of fully hierarchical proxy signatures with warrants, supporting chains of several levels of delegation. Li et al. have proposed their generalization of proxy signature schemes. However, all of Li et al.'s schemes have a common security weakness. In Li et al.'s schemes, an adversary first intercepts a valid proxy signature generated by a proxy group. From the intercepted proxy signature, the adversary can forge illegal proxy signatures being likely generated by the proxy group on behalf of an adversary. To overcome this weakness, in [27] an improvement is proposed. With the proxy key, the proxy signer can sign messages on behalf of the original signer. In cases where the proxy signer misuses the delegated rights, the original signer needs to revoke the proxy signer's signing capability. Currently, the proxy revocation protocols have two approaches. One approach is to change the public key of the original signer. This approach is impractical because, once the public key of the original signer is changed, all signatures generated earlier by the original signer can no longer be verified. The other approach is to put proxy information on a public revocation list. Any verifier must ensure that the received signatures do not have proxy information on the list before verification. However, these approaches have two serious drawbacks. One is that, once the proxy information is posted, all valid proxy signatures generated using that information earlier can no longer be verified. The other drawback is that the size of the revocation list will grow unlimitedly. In [23] a proxy blind signature scheme based on Elliptic curve with proxy revocation is proposed. They achieve it by embedding non-blind time stamp in the signatures and thus the original signer can revoke delegation whenever necessary.

In [17], the information sent to the proxy signer is sent through a secure channel and it does not contain the iden-

tity information of the proxy signer. Here the original signer can play the role of the proxy signer as the key sent by the original signer is used as the proxy key. Also, there is no facility to send warrant messages to proxy signer and verifier, the verifier is unable to determine the time of proxy signature generation and also there is no provision for proxy signer revocation. In [16], the original signer could not play the role of the proxy signer but the other drawbacks were same as in [17]. In [11], the concept of partial delegation by warrant was introduced which helped to put a limit on the delegated messages but the other drawbacks were same as in [17]. In the scheme proposed in [15], the original signer could not play the role of the proxy signer and also there was limit on the delegated messages but a secure channel was necessary to send the required information to the proxy signer. Also the information sent to proxy signer did not contain the identity information of the proxy signer. The scheme is unable to determine the time of proxy signature generation and also there is no provision for proxy signer revocation.

In [30], a proxy signature scheme which uses self certified public keys is proposed. In [26], proxy signature schemes based on factoring are discussed. In [31], the authors describe a Proxy signature scheme based on RSA. In [25] a factorisation based forward-secure proxy signature scheme is proposed. The scheme is based on the forward-secure scheme of Abdalla and Reyzin. The authors also present a Forward-secure threshold proxy signature scheme. Remarkable work is also done on proxy multi-signatures. In [28] an identity based proxy multi-signature scheme is proposed and in [29], a proxy multi-signature scheme based on Elliptic curves is proposed.

Initially we propose a simple proxy signature scheme which satisfies the basic requirements of a secure proxy signature scheme. We find that the scheme can be used to control delegation of financial power to a proxy signer. Using our scheme the proxy signer will be able to submit an e-cheque for only the amount he is entitled to by the original signer. Any cheating by the original signer or the proxy signer is identified by the verifier i.e. the bank.

Next we propose two Forward-secure proxy signature schemes, one is based on DSA and the other is based on popular Bellare-Miner scheme. Both the schemes have the the following features in common:

- An original signer can delegate his signing capability to any number of proxy signers in varying time periods.
- Though the original signer gives proxy information to all the proxy signers at the beginning of the protocol, the proxy signers will be able to generate proxy signatures only in their allotted time periods.
- Proxy signer's signature is made Forward-secure. This mitigates the damage caused by key exposure.
- Identity of the proxy signer is available in the information sent by original signer to proxy signer.

- Secure channel is not required to send the information to proxy signer.
- There is a facility to send warrant messages to proxy signer and verifier.
- Original signer cannot play the role of proxy signer.
- Verifier can determine when the proxy signature was generated.
- Proxy revocation is possible by which an original signer can revoke the signing capability given to proxy signer at his will and the verifier can also test whether the signature is from a revoked signer before verifying the proxy signature.

The Bellare-Miner Forward-secure proxy scheme can also be used as a Forward-secure threshold proxy signature scheme. In literature, to the authors knowledge there is no single proxy signature scheme having all the above features.

The organisation of our paper is as follows: In Section 2, we discuss the basic security requirement of any proxy signature scheme. In Section 3, we describe our simple proxy signature scheme. In Section 4 we give the model for Controlled Delegation in e-cheques using Proxy Signatures. In Section 5 we introduce Forward-security and its need in proxy signatures. In Section 6 we give the description of our proxy signature scheme and discuss two new schemes, one based on DSA and the other based on Bellare-Miner scheme. Lastly in Section 7 we conclude. In Appendix we discuss DSA scheme, Forward-secure DSA signature scheme and Forward-secure Bellare-Miner scheme.

II. Security requirements of a Proxy Signature Scheme

Any secure proxy signature scheme should satisfy the following five requirements :

1. Verifiability : From the proxy signature, a verifier is convinced of the original signer's agreement on the signed message.
2. Strong unforgeability : Only the designated proxy signer can create a valid proxy signature on behalf of the original signer.
3. Strong Identifiability : Anyone can determine the identity of the corresponding proxy signer from the proxy signature.
4. 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.
5. Proxy signer's deviation : A proxy signer cannot create a valid signature not detected as a proxy signature.

III. Simple Proxy Signature Scheme

Following is the description of a simple proxy signature scheme which satisfies the basic requirements of a proxy signature scheme.

A. Proxy key generation:

Let p, q be two large primes such that $q|(p-1)$ and $G_q = \langle g \rangle$ is a q -order multiplicative subgroup of Z_p^* generated by an element $g \in Z_p^*$.

The original signer is Alice 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 $y_B = g^{x_B} \bmod p$.

Alice chooses a random number $k_A \in Z_q^*$, computes

$$K = g^{k_A} \bmod p \quad (1)$$

$$S_A = k_A \cdot y_B + x_A \cdot h(ma) \quad (2)$$

where h is a collision resistant hash function and ma is the message. As both the secret key, x_A , of the original signer(Alice) and the public key, y_B , of the proxy signer(Bob) is used to calculate S_A , Alice cannot deny that Bob is the proxy signer.

Then the tuple (ma, K, S_A) is sent to the proxy signer Bob, who checks its validity by

$$g^{S_A} \equiv y_A^{h(ma)} \cdot K^{y_B} \bmod p. \quad (3)$$

Notice that since

$$\begin{aligned} RHS &= y_A^{h(ma)} \cdot K^{y_B} \bmod p \\ &= g^{x_A \cdot h(ma)} \cdot g^{k_A \cdot y_B} \bmod p \\ &= g^{x_A \cdot h(ma) + k_A \cdot y_B} \bmod p \\ &= g^{S_A} \\ &= LHS \end{aligned}$$

the tuple (ma, K, S_A) sent by an honest signer will be accepted.

If the above verification is correct, Bob sets his proxy key pair (x_p, y_p) as follows :

$$x_p = S_A + x_B \cdot y_A \bmod p \quad (4)$$

$$y_p = g^{x_p} \bmod p \quad (5)$$

B. 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 message m . The resulting proxy signature is the tuple $(sign(m, x_p), K, m, y_A, y_B)$.

This signature also helps to identify the original signer and the proxy signer. Once the verification of this signature for a given message passes with the computation of proxy public

key given by equation (6), the identity of the original signer and the proxy signer is confirmed. Thus the third requirement, Strong identifiability, of a secure proxy signature is satisfied.

We observe in equation(4) that the proxy private key x_p used to generate the signature is computed using the private key of the proxy signer and the public key of the original signer. Thus the proxy signer is creating a valid proxy signature on behalf of the original signer. He therefore cannot repudiate the signature against anyone else. Thus the fourth requirement, Strong undeniability, of a secure proxy signature is satisfied.

C. Proxy signature verification:

The verifier computes the proxy public key y_p as follows :

$$y_p = y_A^{h(m)} \cdot K^{y_B} \cdot y_B^{y_A} \pmod p \quad (6)$$

We observe that the above computation also corresponds to the same public key computed by the proxy signer i.e.

$$\begin{aligned} LHS &= g^{x_p} \pmod p \\ &= g^{S_A + x_B \cdot y_A} \pmod p \\ &= g^{x_A \cdot h(ma) + k_A \cdot y_B + x_B \cdot y_A} \pmod p \\ &= g^{x_A \cdot h(ma)} \cdot g^{k_A \cdot y_B} \cdot g^{x_B \cdot y_A} \pmod p \\ &= y_A^{h(ma)} \cdot K^{y_B} \cdot y_B^{y_A} \pmod p \\ &= RHS. \end{aligned}$$

Finally, the verifier checks whether $(\text{sign}(m, x_p))$ is a valid signature of message m with respect to the proxy public key y_p (given by equation(6)) in the corresponding DLP (Discrete Log Problem) based signature scheme. If this check passes, the verifier is convinced of the original signer's agreement on the signed message as the public key used to verify the signature is calculated using the public key and the security parameter K of original signer. Thus the first requirement, Verifiability, of a secure proxy signature is satisfied.

Also, the proxy signature is identified as a proxy signature and not as an ordinary signature as it is verified only by the proxy public key (y_p) and not by the public key of the proxy signer (y_B). Thus the fifth requirement, that a proxy signer cannot create a valid proxy signature not detected as a proxy signature, of a secure proxy signature is satisfied.

D. Security of our scheme

1. Forgery by the Original Signer: The original signer can generate the proxy public key using equation (6). But he cannot generate the proxy private key x_p from y_p as it amounts to solving a discrete log problem given by equation (5). Thus the original signer is unable to sign like the proxy signer. Therefore forgery by original signer is computationally not possible.

2. Impersonating attack: Let us assume that Bob is not designated as a proxy signer by the original signer Alice. Though Bob can generate a proxy key pair (x'_p, y'_p) with K' and m' satisfying equation (6) and sign a message on behalf of Alice, the verifier who also computes the proxy public key y_p using (K, m) sent by original signer will be able to identify that the proxy public key y_p is not equal to y'_p . By this the verification equation of the DLP based signature scheme fails. Thus Bob cannot become the proxy signer unless he is designated by the original signer Alice.
3. Framing attack: In this attack, a third party Charlie forges a proxy private key and then generates valid proxy signatures such that the verifier believes that these proxy signatures were signed by the proxy signer Bob on behalf of the original signer Alice. When such a proxy signature is presented, Alice cannot deny that she is the original signer of the proxy signer Bob. The result is that Alice and Bob will be framed.

To accomplish this attack, Charlie needs to generate Bob's proxy key pair (x_p, y_p) with K and m satisfying equation (6). y_p is not publicly announced by the proxy signer, Bob, but instead computed by the verifier just before verification. Even if this key is made available, Charlie cannot generate the proxy private key as it is a discrete log problem given by equation (5).

Thus our scheme withstands the above attacks. By this we can say that 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 proxy signer cannot create a valid signature. Thus the second requirement, Strong unforgeability, of a secure proxy signature is satisfied.

IV. Controlled Delegation in e-cheques using Proxy Signatures

We have considered the scenario of organising a conference. The chairman is given the financial power to distribute the funds to various committees. Generally the following methods are used:

On the requisition (based on budget) of the committee members

1. The chairman gives cheques in the name of the committee member for a specified amount.
2. The chairman transfers the amount to the account of committee member. We have come up with a scheme in which the chairman can delegate his signing capability to the committee members who use proxy signatures to sign e-cheques. Here chairman has a controlled delegation i.e. the chairman decides for what amount each member is entitled to spend. The member can only draw the amount for which he is entitled to. The model is shown in figure (1).

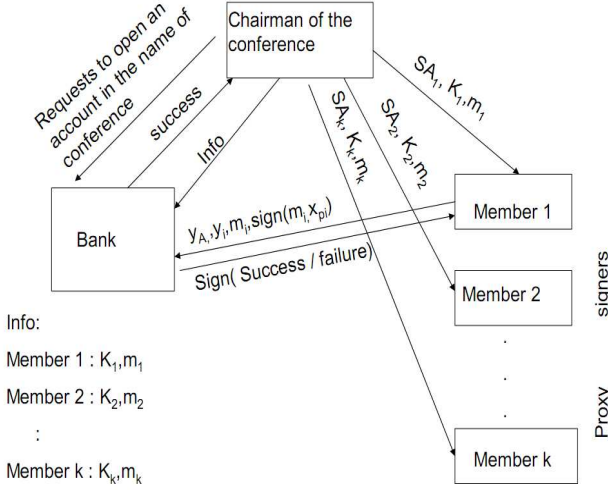


Figure. 1: Model for e-cheque processing with proxy signers

The chairman requests the bank to open an account in the name of the conference and puts all the funds in that account. The chairman sends the signed security parameter K_i and the amount that each member is entitled to spend m_i to the bank. He also sends the signed triplet (SA_i, K_i, m_i) to each of the i^{th} member. K_i and SA_i are computed as follows :

$$K_i = g^{k_{Ai}} \text{ mod } p \quad (7)$$

$$SA_i = k_{Ai} \cdot y_i + x_A \cdot h(m_i) \quad (8)$$

where $k_{Ai} \in \mathbb{Z}_q^*$. y_A is the public key and x_A is the private key of the of the chairman(original signer) . y_i is the public key and x_i is the private key of the i^{th} member.

Each member checks the validity of the information received from the chairman using the following equation

$$g^{SA_i} \equiv y_A^{h(m_i)} \cdot K_i^{y_i} \text{ mod } p. \quad (9)$$

Each member creates a proxy key pair using the following equations:

$$x_{pi} = SA_i + x_i \cdot y_A \text{ mod } p \quad (10)$$

$$y_{pi} = g^{x_{pi}} \text{ mod } p \quad (11)$$

and signs the e-cheque for the amount m_i using the proxy secret key x_{pi} . The signature scheme can be any discrete log based scheme. We have chosen the widely used DSA scheme. The proxy signature is $(sign(m_i, x_{pi}), m_i, K_i, y_A, y_B)$. $sign(m_i, x_{pi})$ is generated as in DSA (see Appendix A). The remaining parameters specified in the signature help the verifier(bank) to compute the proxy public key y_{pi} . In other words each of the i^{th} member requests the verifier to verify the signature using the proxy public key y_{pi} and not his public key y_i .

On receiving the proxy signature the bank computes the proxy public key using the following equation :

$$y_{pi} = y_A^{h(m_i)} \cdot K_i^{y_i} \cdot y_i^{y_A} \text{ mod } p \quad (12)$$

The signature is verified using the above proxy public key with the help of the verification equation (see Appendix A) of DSA scheme.

If there is change in m_i i.e. the amount for which the proxy signer is entitled to, or in any other parameters, the proxy public key computed by proxy signer will be different from that computed by the verifier. By this the verification equation of the DSA scheme will not hold good and the cheating is identified.

V. Forward Security and its need in Proxy Signatures

Digital signatures are vulnerable to leakage of secret key. If the secret key is compromised, any message can be forged. To prevent future forgery of signatures, both public key and secret key must be changed. Notice, that this will not protect previously signed messages: such messages will have to be re-signed with new pair of public key and secret key, but this is not feasible. Also changing the keys frequently is not a practical solution.

To address the above problem, the notion of forward security for digital signatures was first proposed by Anderson in [1], and carefully formalised by Bellare and Miner in [6] (see also [2, 3, 9, 12, 13]). The basic idea is to extend a standard digital signature scheme with a key updation algorithm so that the secret key can be changed frequently while the public key stays the same. Unlike a standard signature scheme, a forward secure signature scheme has its operation divided into time periods, each of which uses a different secret key to sign a message. The key updation algorithm computes the secret key for the new time period based on the previous one using a one way function. Thus, given the secret key for any time period, it is hard to compute any of the previously used secret keys. (It is important for the signer to delete the old secret key as soon as the new one is generated, since otherwise an adversary breaking the system could easily get hold of these undeleted keys and forge signatures.) Therefore a receiver with a message signed before the period in which the secret key gets compromised, can still trust this signature, for it is still hard to any adversary to forge previous signatures.

To specify a forward-secure signature scheme, we need to (i) give a rule for updating the secret key (ii) specify the public key and (iii) specify the signing and the verification algorithms.

As digital signatures, proxy signatures are also vulnerable to leakage of proxy secret key. If the proxy secret key is compromised, any message can be forged. To prevent future forgery of signatures, both proxy public key and proxy secret key must be changed which forces the original signer to change the proxy information. But this will not protect

previously signed messages: such messages will have to be re-signed with new pair of proxy public key and secret key which is not feasible. To address this problem, we need the concept of forward-security for proxy signatures.

VI. Forward-Secure Proxy Signature Schemes for multiple proxy signers with Proxy Revocation

The basic idea behind the construction of our scheme is as follows: Suppose Alice wants to delegate her signing power to a set of l proxy signers $\{P_1, P_2, \dots, P_l\}$. Alice in the role of original signer divides the time interval into equal periods T_i , where $i = 1, 2, \dots, k$. Now, Alice allocates time periods to proxy signers by finding a map from the set $\tau = \{T_1, T_2, \dots, T_k\}$ to $\varrho = \{P_1, P_2, \dots, P_l\}$.

If $k < l$, then a time period T_i will be allocated to more than one proxy signer.

If $k > l$, then a proxy signer P_j will be allocated more than one distinct time period T_i .

If $k = l$, then each proxy signer P_j will be allocated one time period.

Alice computes part of the proxy information $SA_{i,j}$ for each of the proxy signers P_j for the allotted time period T_i . $SA_{i,j}$ is sent along with other proxy information to the proxy signer P_j . Though Alice gives proxy information to all the proxy signers at the beginning of the protocol, the proxy signers will be able to generate proxy signatures only in their allotted time periods. Each of the proxy signers verify that Alice has designated him/her as proxy signer. If the verification holds the proxy signers compute the proxy key pair (x_{p0}, y_p) . They divide the time period i into T' time periods and sign any messages in these i' time periods (which ranges from 1 to T') using Forward-secure signatures (see Figure 1). Thus the proxy signatures generated by proxy signers are Forward-secure proxy signatures.

In case Alice wants to revoke any of the proxy signers, she enters their public key in the common proxy revocation list and sends a revocation message to the proxy signer. This entry exists in the proxy revocation list until the expiry of the allotted time period for that proxy signer and later deleted. By deleting the entries at the right time period, the proxy revocation list is kept as small as possible. Even after receiving the revocation message if a proxy signer tries to create proxy signatures using available or self generated proxy information, the verifier rejects the signature. See Figure 2 for the Proxy Signature Model with Proxy Revocation.

On receiving the proxy signatures, the verifier first checks whether the public key of the proxy signer is available in the proxy revocation list. If it is available, the signature is rejected, else the verifier follows a two step procedure to verify the signature. In the first step, the verifier verifies an equation whose success indicates that there is an agreement between the original signer and the proxy signer on the signed message. In the second step the verifier verifies the signa-

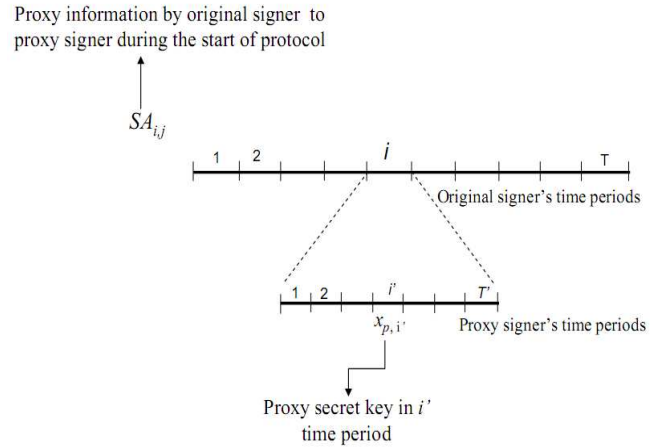


Figure. 2: Basic idea of the scheme

ture using the verification equation of the standard signature scheme (here DSA / Bellare-Miner Signature scheme). Only if equations in both the steps satisfy, the verifier accepts the received signature.

The special feature of our scheme is that an original signer can delegate his signing capability to any number of proxy signers in varying time periods. Further, the original signer gives proxy information to all the proxy signers at the beginning of the protocol, the proxy signers will be able to generate proxy signatures only in their allotted time periods. Therefore, there is no intervention of original signer after she distributes the proxy information to all proxy signers in their allotted time period. In existing schemes the power to sign must be delegated only in the allotted time period i.e. the original signer is required to send proxy information to the proxy signers in their allotted time period. There is no way to enable the proxy signer to generate proxy signatures only in specified time periods after delegating the power to sign. Thus in our scheme the original signer is relieved from sending proxy information to proxy signers in their period of generating proxy signatures.

Let us assume that there are P_l proxy signers for the original signer Alice. Let us consider the situation of P_2 who is the proxy signer in time periods $i = 2$ and $i = 3$. In Figure 3.a., the original signer sends the proxy information to all the proxy signers in time period $i = 1$. This time period i is divided into T' time periods and in the first time period i.e. $i' = 1$, the proxy signer P_2 generates Forward-secure proxy signatures on message m and sends it along with other information to verifier. The verifier first checks in the proxy revocation list whether the public key y_2 of P_2 exists. Let us assume that y_2 does not exist in the list i.e. the original signer has not revoked the proxy signer P_2 . The verifier then, performs Step 1 of proxy signature verification and verifies whether the signature is from a valid proxy signer. Here the

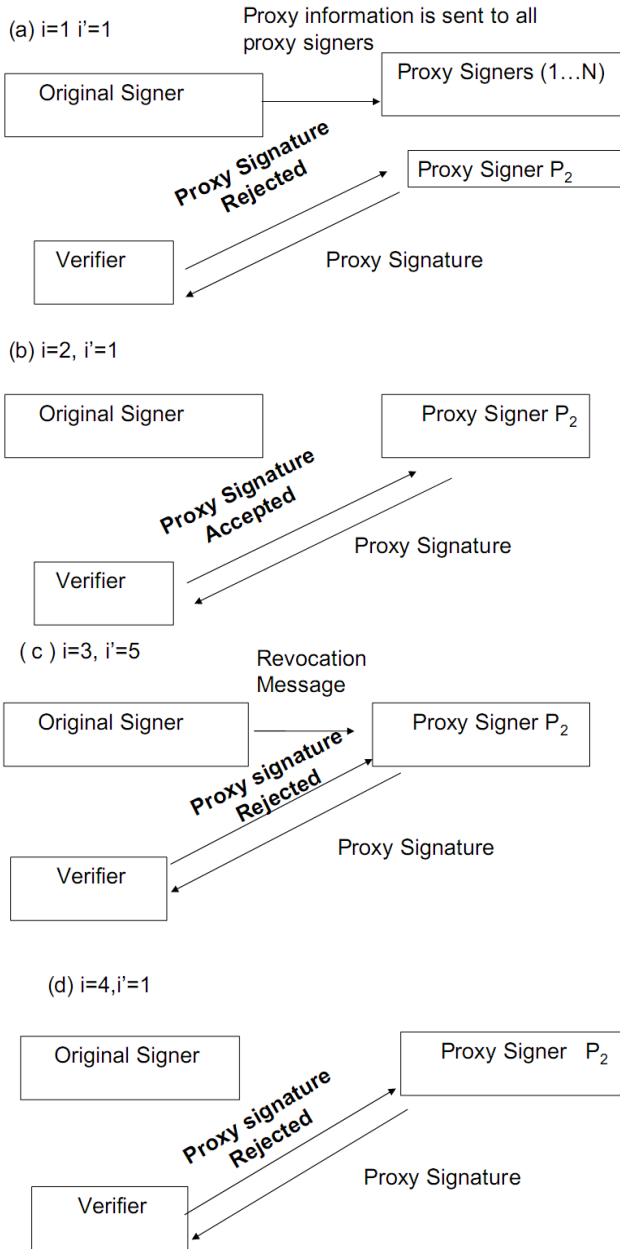


Figure. 3: Forward-Secure Proxy Signature Model with Proxy Revocation

verification fails as the proxy signer does not possess proxy information of time period $i = 1$ and thus the signature generated is invalid. Therefore the verifier rejects the proxy signature. In Figure 3.b., in time period $i = 2$ & $i' = 1$, the proxy signer generates Forward-secure proxy signature on message m using the proxy information received earlier and sends it to the verifier for acceptance. Now the signature is accepted as the signature is generated using the proxy information of time period $i = 2$. In Figure 3.c. we observe the communication among the players of the proxy signature model in time period $i = 3$ & $i' = 5$. Here let us assume that the original signer has entered the public key of the proxy signer in the proxy revocation list and has sent a revocation message to the proxy signer. Now the proxy signer is a revoked signer. If the proxy signer tries to generate a proxy signature using proxy information available with him for time period $i = 3$, the signature is rejected by the verifier. In Figure 3.d. we observe the working of the model in time period $i = 4$ & $i' = 1$. If the proxy signer generates a proxy signature using previous time period proxy information or self generated proxy information, the verifier rejects the signature in the Step 1 of proxy signature verification as the proxy signer is automatically revoked after the completion of his allotted time periods.

Proxy Revocation: The proxy signers may misuse the power to sign given by the original signer. In such cases, Alice may decide to revoke the proxy signer. This can be conveniently done by Alice by entering the public key of the malicious proxy signer in the common proxy revocation list. Also, a revocation message is sent to the concerned proxy signer. This entry exists in the proxy revocation list until the expiry of the allotted time period of that proxy signer and later deleted. By deleting the entries at the right time period, the proxy revocation list is kept as small as possible. Even after receiving the revocation message if a proxy signer tries to create proxy signatures using available or self generated proxy information, the verifier rejects the signature. Also, automatic proxy revocation is provided i.e. immediate revocation after the expiry of the allotted time period of the proxy signer.

A. Forward-secure proxy signature scheme for multiple proxy signers using DSA with proxy revocation

Following is the protocol for Forward-secure proxy signature scheme for multiple proxy signers using DSA:

1) Initial Setup

Let p, q be two large primes such that $q|(p - 1)$ and $G_q = \langle g \rangle$ is a q -order multiplicative subgroup of Z_p^* generated by an element $g \in Z_p^*$. The certified key pair of the original signer is (x_A, y_A) . The secret key x_A is chosen randomly in the range $1 < x_A < p - 1$. The public key is given by $y_A = g^{x_A} \text{ mod } p$.

Similarly, each of the proxy signer P_j has a certified key pair

(x_j, y_j) where $y_j = g^{x_j} \bmod p$.

2) Proxy Generation

Alice wants P_j , $j = 1, 2, \dots, l$ where l is the number of proxy signers, to be the proxy signers in different time periods T_i (where $1 < i \leq k$).

Alice computes $K_{i,j}$ and $SA_{i,j}$ for each of the proxy signer P_j , for the allotted time period T_i . By choosing a random number $k_{i,j} \in Z_q^*$ she computes

$$K_{i,j} = g^{k_{i,j}} \bmod p. \quad (13)$$

$$SA_{i,j} = k_{i,j} \cdot y_j \cdot i + x_A \cdot h(ma || y_j) \quad (14)$$

where, h is a collision resistant hash function and ma is the message for proxy signer and the verifier (which may include warrants).

3) Proxy Delivery

The proxy information sent to proxy signer P_j for the allocated time period T_i consists of the tuple $(ma, i, K_{i,j}, g^{SA_{i,j}}, y_A, y_j)$.

4) Proxy Verification

Each proxy signer P_j confirms his role as proxy signer during the time period T_i by checking the validity of the following equation:

$$g^{SA_{i,j}} \equiv y_A^{h(ma || y_j)} \cdot K_{i,j}^{i \cdot y_j} \bmod p. \quad (15)$$

Notice that since

$$\begin{aligned} RHS &= y_A^{h(ma || y_j)} \cdot K_{i,j}^{i \cdot y_j} \bmod p \\ &= g^{x_A \cdot h(ma || y_j)} \cdot g^{k_{i,j} \cdot i \cdot y_j} \bmod p \\ &= g^{x_A \cdot h(ma || y_j) + k_{i,j} \cdot i \cdot y_j} \bmod p \\ &= g^{SA_{i,j}} \bmod p \\ &= LHS \end{aligned}$$

the tuple $(ma, K_{i,j}, g^{SA_{i,j}}, y_A, y_j)$ sent by an honest signer will be accepted.

5) Proxy Key Generation

If the above verification is correct, each proxy signer P_j sets his initial proxy secret x_{p0} as

$$x_{p0} = (SA_{i,j} + x_j \cdot y_A)^{kj} \bmod p \quad (16)$$

where kj is a random number chosen such that $0 < kj < p$. The public key (y_p) is computed as

$$y_p = \mathcal{A}(g, T', x_{p0}) \bmod p \quad (17)$$

where T' is the number of sub time periods in time period i . Proxy secret key $x_{pi'}$ for any time period i' (i' ranges from 1 to T') is obtained by updating the secret key $x_{p(i'-1)}$ of the previous time period via the update rule

$$x_{pi'} = g^{x_{p(i'-1)} \bmod \phi^{T-i+1}(p)} \bmod \phi^{T-i}(p)$$

We observe in equation (16) that the proxy private key x_{p0} used to generate the signature is computed using the private key x_j of the proxy signer P_j and the public key y_A of the original signer. Thus, the proxy signer is creating a valid proxy signature on behalf of the original signer. He therefore cannot repudiate his signature later. Thus the fourth requirement, Strong undeniability, of a secure proxy signature is satisfied.

6) Proxy signature generation:

The proxy signer P_j uses DSA Forward-secure signature scheme to generate proxy signature on any message m for the allocated time period T_i and subtime period $T_{i'}$, where $i = 1, \dots, T$, $i' = 1, \dots, T'$. The signature $((r, s), m, ma, i, i', g^{SA_{i,j}}, K_{i,j}, y_p)$ is generated as follows:

- A random value k per message is generated where $0 < k < q$.
- $r = (g^k \bmod p) \bmod q$
- $s = k^{-1}(\text{SHA}(m || i') + (\mathcal{A}(g, T' - i' - 1, x_{pi'})) \cdot r) \bmod q$ where $\text{SHA}(m)$ is the SHA hash function applied to the message m

7) Proxy signature verification:

The verifier receives the signature $((r, s), m, ma, i, i', g^{SA_{i,j}}, K_{i,j}, y_p, y_j, y_A)$ for the proxy signer P_j . He first checks whether the public key of the proxy signer is available in the proxy revocation list. If it is available, the signature is rejected, else the verifier follows the following two step procedure to verify the signature.

Step1: In this step the verifier checks whether the signature is from a valid proxy signer P_j of the original signer. By verifying the following equation (same equation as used by P_j to confirm his role as proxy signer), the verifier is convinced that there is an agreement between the original signer and the proxy signer on the signed message. Thus the first requirement, Verifiability, of a secure proxy signature is satisfied. Also, the verifier is convinced that the signature is not from a revoked signer.

$$g^{SA_{i,j}} \equiv y_A^{h(ma || y_j)} \cdot K_{i,j}^{i \cdot y_j} \bmod p. \quad (18)$$

Step2: If the above equation holds the verifier verifies the received signature as follows:

$$w = (s)^{-1} \bmod q$$

$$\begin{aligned}
u_1 &= (SHA(m||i').w) \bmod q \\
u_2 &= r.w \bmod q \\
v &= ((g^{u_1} * y_p^{u_2}) \bmod p) \bmod q
\end{aligned}$$

The signature is valid if

$$v = r. \quad (19)$$

The time period, i' , in the signature informs the verifier when the proxy signature is generated.

Notice that

$$\begin{aligned}
LHS &= g^{u_1} * y_p^{u_2} \\
&= g^{SHA(m)*w} * y_p^{r*w} \\
&= g^{w*(SHA(m)+A(g,T'-i',x_{pi'})*r)} \bmod p \\
&= g^{s^{-1}(SHA(m)+A(g,T'-i',x_{pi'})*r)} \bmod p \\
&= g^k \\
&= r \\
&= RHS
\end{aligned}$$

Therefore, if both the equations (18) and (19) are satisfied, then the signature sent by an honest proxy signer will be accepted.

In our proxy signature scheme, we slightly deviate from the basic working of most proxy signature schemes. Here the proxy signer sends the proxy public key to the verifier along with the proxy signature while in most other schemes the proxy public key is computed by the verifier. The purpose of making the verifier to compute the proxy public key is to confirm that there is an agreement between the original signer and the proxy signer on the signed message. But this confirmation is obtained from the Step 1 of our proxy signature verification. Therefore, we avoid the public key computation by the verifier. Also, for any proxy key pair (x_{pi}, y_p) , the proxy signer can satisfy the equation (19), provided he is identified as a valid proxy signer in Step 1. Any proxy signature verified in Step 2 is associated with proxy signer P_j whose public key is y_j and original signer whose public key is y_A .

Also, the signature is identified as a proxy signature and not as an ordinary signature as it is verified only by the proxy public key (y_p) and not by the public key of the proxy signer (y_B). Thus the fifth requirement, that a proxy signer cannot create a valid proxy signature not detected as a proxy signature, of a secure proxy signature is satisfied.

This proxy signature also helps to identify the original signer and the proxy signer. Once the verification of this signature for a given message passes both Step 1 and Step 2, the identity of the original signer and the proxy signer is confirmed. Thus the third requirement, Strong identifiability, of a secure proxy signature is satisfied.

8) Security of our scheme

1. Forgery by the Original Signer: The proxy secret key is dependent on both the proxy information sent by the original signer as well as the secret key of the proxy signer. Recovering proxy secret key $x_{pi'}$ from proxy public key y_p is equivalent to solving the following set of equations:

$$y_{pl} = g_l^{g_l^{(l+1)}} \bmod \phi^l(p)$$

for $l = 0, 1, \dots, (T' - 2)$.

$$y_{pl} = g_l^{g_0} \bmod \phi^l(p)$$

for $l = (T' - 1)$. g_0 is the initial secret key x_{p0} of the proxy signer. As T' is chosen in such a way that $\phi^l(p)$ has a large prime factor, solution of the above set of equations is computationally infeasible. Therefore, the original signer cannot derive the proxy secret key from the proxy public key and thus cannot sign like the proxy signer. Hence, forgery by original signer is computationally not possible.

2. Impersonating attack: Let us assume that Bob is not designated as a proxy signer by the original signer Alice. Though Bob can generate a proxy key pair (x'_p, y'_p) with K' and m' satisfying equations (16 & 17) and sign a message on behalf of Alice, the verifier on receiving the signatures, in the Step 1 of Proxy Signature Verification, checks whether the signature is from a valid proxy signer or not. During this test if the verification fails, the verifier rejects the signature. Thus Bob cannot become the proxy signer unless he is designated by the original signer Alice.
3. Framing attack: In this attack, a third party Charlie forges a proxy private key and then generates valid proxy signatures such that the verifier believes that these proxy signatures were signed by the proxy signer Bob on behalf of the original signer Alice. When such a proxy signature is presented, Alice cannot deny that she is the original signer of the proxy signer Bob. The result is that Alice and Bob will be framed. To accomplish this attack, Charlie needs to forge Bob's proxy key pair (x_p, y_p) . As forward-secure signatures are used by proxy signer it is computationally difficult to forge the proxy secret key [1, 2, 6, 9, 12, 13]. Knowing the proxy public key y_p Charlie cannot generate the proxy private key as discussed under the topic Forgery by the Original Signer in the current section.

Thus our scheme withstands the above attacks. By this we can say that 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 proxy signer cannot create a valid signature. Thus the second requirement, Strong unforgeability, of a secure proxy signature is satisfied.

B. Bellare-Miner Forward-secure Proxy signature scheme with Proxy revocation

Following is the protocol for Bellare-Miner Forward-secure Proxy signature scheme:

1) Initial Setup

Let p, q be two large primes each congruent to 3 mod 4. Let $N = p \cdot q$

The initial secret key of Alice is $SK_{A,0} = (SK_{A,(1,0)}, \dots, SK_{A,(l,0)}, N, 0)$ where $SK_{A,(k,0)} \stackrel{R}{\leftarrow} Z_N^*$, where $k = 1, \dots, l$.

Secret key $SK_{A,i} = (SK_{A,(1,i)}, \dots, SK_{A,(l,i)})$ for any time period i is obtained by updating the secret key $SK_{A,i-1} = (SK_{A,(1,i-1)}, \dots, SK_{A,(l,i-1)})$ of the previous time period via the update rule

$$SK_{A,(k,i)} = SK_{A,(k,i-1)}^2 \pmod{N}, \quad (20)$$

where $k = 1, \dots, l$.

The public key of Alice is $U_A = (U_{A,1}, \dots, U_{A,l})$, is calculated as the value obtained on updating the base secret key $T + 1$ times:

$$U_{A,k} = SK_{A,(k,0)}^{2^{T+1}} \pmod{N} \quad (21)$$

Proxy Signer P'_j 's initial secret key is $SK_{j,0} = (SK_{j,(1,0)}, \dots, SK_{j,(l,0)}, N_j, 0)$ where $SK_{j,(k,0)} \stackrel{R}{\leftarrow} Z_{N_j}^*$, where $k = 1, \dots, l$. We assume that N_j is less than N .

Secret key $SK_{j,i} = (SK_{j,(1,i)}, \dots, SK_{j,(l,i)})$ for any time period i is obtained by updating the secret key $SK_{j,i-1} = (SK_{j,(1,i-1)}, \dots, SK_{j,(l,i-1)})$ of the previous time period via the update rule

$$SK_{j,(k,i)} = SK_{j,(k,i-1)}^2 \pmod{N_j}, \quad (22)$$

where $k = 1, \dots, l$.

The public key $U_j = (U_{j,1}, \dots, U_{j,l})$ of any proxy signer P_j , is calculated as the value obtained on updating the base secret key $T + 1$ times:

$$U_{j,k} = SK_{j,(k,0)}^{2^{T+1}} \pmod{N_j} \quad (23)$$

2) Proxy Generation

Alice wants P_j , $j = 1, 2, \dots, n$ where n is the number of proxy signers, to be the proxy signers in different time periods T_i (where $1 < i \leq k$).

Alice computes $Y_{i,j}$ and $SA_{i,j}$ for each of the proxy signer P_j , for the allotted time period T_i . By choosing a random number $R_{i,j} \in Z_N^*$ she computes

$$Y_{i,j} = R_{i,j}^{2^{(T+1-i)}} \pmod{N} \quad (24)$$

$$SA_{i,j} = R_{i,j} \cdot \prod_{k=1}^l U_{j,k} \cdot \prod_{k=1}^l SK_{A,(k,i)}^{c_k} \pmod{N} \quad (25)$$

where $c_1, \dots, c_k \leftarrow H(M_w, Y, U_B, j)$, H is a collision resistant hash function and M_w is the message for proxy signer and the verifier (which may include warrants).

3) Proxy Delivery

Alice delegates her signing capability to proxy signer P_j by giving the following information, $(i, M_w, Y_{i,j}, SA_{i,j}, U_A, U_j)$.

4) Proxy Verification

Each proxy signer P_j confirms his role as proxy signer during the time period T_i by checking the validity of the following equation:

$$SA_{i,j}^{2^{(T+1-i)}} = Y_{i,j} \cdot \prod_{k=1}^l U_{A,k}^{c_k} \cdot \prod_{k=1}^l U_{j,k}^{2^{(T+1-i)}} \pmod{N} \quad (26)$$

Notice that since

$$\begin{aligned} LHS &= (R_{i,j} \cdot \prod_{k=1}^l U_{j,k} \cdot \prod_{k=1}^l (SK_{A,(k,i)}^{c_k})^{2^{(T+1-i)}} \pmod{N} \\ &= R_{i,j}^{2^{(T+1-i)}} \cdot \prod_{k=1}^l U_{j,k}^{2^{(T+1-i)}} \cdot \prod_{k=1}^l SK_{A,(k,i)}^{c_k \cdot 2^{(T+1-i)}} \pmod{N} \\ &= Y_{i,j} \cdot (\prod_{k=1}^l U_{j,k})^{2^{(T+1-i)}} \cdot (\prod_{k=1}^l SK_{A,(k,0)}^{c_k \cdot 2^i})^{2^{(T+1-i)}} \pmod{N} \\ &= Y_{i,j} \cdot (\prod_{k=1}^l U_{j,k})^{2^{(T+1-i)}} \cdot (\prod_{k=1}^l U_{A,k}^{c_k}) \pmod{N} \\ &= RHS, \end{aligned}$$

the proxy signer P_j accepts the tuple $(i, M_w, Y_{i,j}, SA_{i,j}, U_A, U_j)$ as valid proxy sent by an honest signer.

5) Proxy Key Generation

If the above verification is correct, each proxy signer P_j sets his initial proxy secret key $x_{p,0} = (x_{p,(1,0)}, \dots, x_{p,(l,0)}, N_j, 0)$ in any time period i as

$$x_{p,(k,0)} = SA_{i,j} \cdot SK_{j,(k,0)} \cdot U_{A,k} \pmod{N_j} \quad (27)$$

where $k = 1, \dots, l$ and N_j is Blum William's integer.

Proxy secret key $x_{p,i'} = (x_{p,(1,i')}, \dots, x_{p,(l,i')})$ for any time period i' is obtained by updating the proxy secret key $x_{p,i'-1} = (x_{p,(1,i'-1)}, \dots, x_{p,(l,i'-1)})$ of the previous time period via the update rule

$$x_{p,(k,i')} = x_{p,(k,i'-1)}^2 \pmod{N_j}, \quad (28)$$

where $i' = 1, \dots, l$.

The proxy public key is $y_p = (y_{p,1}, \dots, y_{p,l})$, is calculated as the value obtained on updating the initial proxy secret key $T + 1$ times:

$$y_{p,k} = x_{p,(k,0)}^{2^{T'+1}} \bmod N_j \quad (29)$$

where T' is the number of sub time periods in time period i . Note that in equation (27) the proxy private key used to generate the proxy signature is computed using the private key of the proxy signer and the public key of the original signer. This ensures that the proxy signer is creating a valid proxy signature on behalf of the original signer. He therefore cannot repudiate his signature. Thus the fourth requirement Strong undeniability, of a secure proxy signature is satisfied.

6) Proxy signature generation:

The proxy signer P_j uses Bellare-Miner Forward-secure signature scheme to generate proxy signature on any message m . The signature in time period i' , $((Y_p, Z_p), m, i, i', y_p, SA_{i,j}, M_w, U_A, U_j, N_j)$ is generated as follows:

The proxy secret key is $x_{p,i'}$.

$$Y_p = R_p^{2^{(T+1-i')}} \bmod N_j \quad (30)$$

where $R_p \xleftarrow{R} Z_N^*$

$$Z_p = R_p \cdot \prod_{k=1}^l x_{p,(k,i')}^{c_k} \bmod N_j \quad (31)$$

where $c_1, \dots, c_l \leftarrow H(m, Y_p, i')$ and H is a collision resistant hash function.

7) Proxy signature verification:

The verifier receives the signature $((Y_p, Z_p), m, i, i', y_p, SA_{i,j}, M_w, U_A, U_j, N_j)$ during time period i' for the proxy signer P_j . He first checks whether the public key of the proxy signer is available in the proxy revocation list. If it is available, the signature is rejected, else the verifier follows the following two step procedure to verify the signature.

Step1: In this step the verifier checks whether the signature is from a valid proxy signer P_j of the original signer. By verifying the following equation (same equation as used by P_j to confirm his role as proxy signer), the verifier is convinced that there is an agreement between the original signer and the proxy signer on the signed message. Thus the first requirement, Verifiability, of a secure proxy signature is satisfied. Also, the verifier is convinced that the signature is not from a revoked signer.

$$SA_{i,j}^{2^{(T+1-i)}} = Y_{i,j} \cdot \prod_{k=1}^l U_{A,k}^{c_k} \cdot \prod_{k=1}^l U_{j,k}^{2^{(T+1-i)}} \bmod N \quad (32)$$

Step2: If the above equation holds the verifier verifies the received signature as follows:

$$Z_p^{2^{(T'+1-i')}} = Y_p \cdot \prod_{k=1}^l (y_{p,k})^{c_k} \bmod N_j \quad (33)$$

Notice that

$$\begin{aligned} LHS &= (R_p \cdot \prod_{k=1}^l x_{p,(k,i')}^{c_k})^{2^{(T'+1-i')}} \bmod N_j \\ &= R_p^{2^{(T'+1-i')}} \cdot (\prod_{k=1}^l x_{p,(k,i')}^{c_k})^{2^{(T'+1-i')}} \bmod N_j \\ &= Y_p \cdot \prod_{k=1}^l (x_{p,(k,0)}^{c_k})^{2^i \cdot 2^{(T'+1-i')}} \bmod N_j \\ &= Y_p \cdot \prod_{i=1}^l y_{p,k}^{c_k} \bmod N_j \\ &= RHS. \end{aligned}$$

Therefore, if both the equations (32) and (33) are satisfied, then the signature sent by an honest proxy signer will be accepted.

In our proxy signature scheme, we slightly deviate from the basic working of most proxy signature schemes. Here the proxy signer sends the proxy public key to the verifier along with the proxy signature while in most other schemes the proxy public key is computed by the verifier. The purpose of making the verifier to compute the proxy public key is to confirm that there is an agreement between the original signer and the proxy signer on the signed message. But this confirmation is obtained from the Step 1 of our proxy signature verification. Therefore, we avoid the public key computation by the verifier. Also, for any proxy key pair (x_{pi}, y_p) , the proxy signer can satisfy the equation (33), provided he is identified as a valid proxy signer in Step 1. Any proxy signature verified in Step 2 is associated with proxy signer P_j whose public key is U_j and original signer whose public key is U_A .

Also, the signature is identified as a proxy signature and not as an ordinary signature as it is verified only by the proxy public key (y_p) and not by the public key of the proxy signer (U_j) . Thus the fifth requirement, that a proxy signer cannot create a valid proxy signature not detected as a proxy signature, of a secure proxy signature is satisfied.

This proxy signature also helps to identify the original signer and the proxy signer. Once the verification of this signature for a given message passes both Step 1 and Step 2, the identity of the original signer and the proxy signer is confirmed. Thus the third requirement, Strong identifiability, of a secure proxy signature is satisfied.

C. Security of our scheme

1. Forgery by the Original Signer: The proxy secret key is dependent on both the proxy information sent by the

original signer as well as the secret key of the proxy signer. Therefore the original signer cannot generate the proxy secret key. He also cannot derive the proxy secret key from the proxy public key given by equation (29) as it is difficult to factorise the Blum William's integer N . Thus the original signer is unable to sign like the proxy signer. Therefore forgery by original signer is computationally not possible.

2. Impersonating attack: Let us assume that Bob is not designated as a proxy signer by the original signer Alice. Though Bob can generate a proxy key pair (x'_p, y'_p) satisfying equations (27,28 and 29) and sign a message on behalf of Alice, the verifier on receiving the signatures, first confirms using a verification equation whether the signature is from a valid proxy signer or from a revoked proxy signer. During this test the verification fails and the verifier considers him as a revoked signer. Thus Bob cannot become the proxy signer unless he is designated by the original signer Alice.
3. Framing attack: In this attack, a third party Charlie forges a proxy private key and then generates valid proxy signatures such that the verifier believes that these proxy signatures were signed by the proxy signer Bob on behalf of the original signer Alice. When such a proxy signature is presented, Alice cannot deny that she is the original signer of the proxy signer Bob. The result is that Alice and Bob will be framed.

To accomplish this attack, Charlie needs to forge Bob's proxy key pair (x_p, y_p) . As forward-secure signatures are used by proxy signer it is computationally difficult to forge the proxy secret key. Knowing the proxy public key y_p Charlie cannot generate the proxy private key given by equation (29) as it is difficult to factorise the Blum William's integer N .

Thus our scheme withstands the above attacks. By this we can say that 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 proxy signer cannot create a valid signature. Thus the second requirement, Strong unforgeability, of a secure proxy signature is satisfied.

VII. Conclusion

We have come up with a simple proxy signature scheme and we use the scheme to have a controlled delegation of financial power to a proxy signer. We have extended our research work on proxy signatures by coming up with two forward secure proxy schemes, one based on Bellare-Miner scheme and the other based on DSA scheme. Forward-Secure proxy signatures guarantee the security of messages signed in the past even if the proxy signer's secret key is exposed today. All our

schemes meet the basic requirements of a proxy signature scheme. The two forward-secure proxy schemes have certain additional properties which make the system more flexible and secure. Here, an original signer can delegate his signing capability to any number of proxy signers in varying time periods. Though the original signer gives proxy information to all the proxy signers at the beginning of the protocol, the proxy signers will be able to generate proxy signatures only in their allotted time periods. The proxy signer is automatically revoked in time periods other than his allotted time period to be a proxy signer. When an original signer identifies that a proxy signer is misusing the signing capability, he can revoke the proxy signer such that all his future signatures are rejected by the verifier during his allotted time periods.

References

- [1] Anderson, R. Invited Lecture, *Fourth Annual Conference on Computer and Communications Security*, ACM, (1997).
- [2] Abdalla, M., Reyzin, L. "A New Forward-Secure Digital Signature Scheme". In *Proceedings of ASIACRYPT 2000*, LNCS 1976, pp. 116-129. Springer-Verlag, (2000), 116-129.
- [3] Michel Abdalla, Sara Miner, Chanathip Namprempre. "Forward-Secure Threshold Signature Schemes". In *Proceedings of Topics in Cryptology CT-RSA 2001*, Volume 2020 of Lectures Notes in Computer Science, San Francisco, CA, USA, Apr. 8-12, 2001. Springer-Verlag, Berlin, Germany.
- [4] A. Boldyreva, A. Palacio, and B. Warinschi. "Secure proxy signature schemes for delegation of signing rights". Available at <http://eprint.iacr.org/2003/096>
- [5] B. Lee, H. Kim, and K. Kim. "Secure mobile agent using strong non-designated proxy signature." In *Proceedings of Information Security and Privacy (ACISP01)*, LNCS 2119, pp.474-486. Springer-Verlag, 2001.
- [6] Bellare, M., Miner, S. "A Forward-Secure Digital Signature Scheme." In *Proceedings of Advances in Cryptology-Crypto 99*, Lecture notes in Computer Science, Vol. 1666. Springer-Verlag, (1999).
- [7] H. Ghodosi and J. Pieprzyk. "Repudiation of cheating and non-repudiation of Zhangs proxy signature schemes." In *Proceedings of Information Security and Privacy (ACISP99)*, LNCS 1587, pp. 129-134. Springer-Verlag, 1999.
- [8] M. Ai-Ibrahim and A. Cerny., "Proxy and threshold one-time signatures". In *Proceedings of the 11th International Conference Applied Cryptography and Network Security (ACNS03)*, LNCS 2846, Springer-Verlag, 2003.

- [9] Itkis, G., Reyzin, L. "Forward-secure signatures with optimal signing and verifying." In *Proceedings of CRYPTO'01*, LNCS 2139, Springer-Verlag, (2001), 332-354.
- [10] J.-Y. Lee, J. H. Cheon, and S. Kim. "An analysis of proxy signatures: Is a secure channel necessary?" In *Proceedings of Topics in Cryptology - CT-RSA 2003*, LNCS 2612, pp. 68-79. Springer-Verlag, 2003.
- [11] S. Kim, S. Park, and D. Won. "Proxy signatures, revisited." In *Proceedings of Information and Communications Security (ICICS97)*, LNCS 1334, pp. 223-232. Springer-Verlag, 1997.
- [12] Krawczyk, H. "Simple forward-secure signatures from any signature scheme". In: *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, ACM, (2000), pp. 108-115.
- [13] Kozlov, A, Reyzin, L. "Forward-Secure Signatures with Fast Key Update." In *Proceedings of Security in Communication Networks (SCN 2002)*, LNCS 2576, Springer-Verlag, pp.241-256.
- [14] K. Zhang. "Nonrepudiable proxy signature schemes". Manuscript 1997 Available at <http://citeseer.nj.nec.com/360090.html>.
- [15] B. Lee, H. Kim, and K. Kim. "Strong proxy signature and its applications". In *Proceedings of the 2001 Symposium on Cryptography and Information Security (SCIS01)*, Vol. 2/2, pp. 603-608.
- [16] 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.
- [17] M. Mambo, K. Usuda, E. Okamoto. "Proxy signatures for delegating signing operation". In *Proceedings of 3rd ACM Conference on Computer and Communications Security (CCS96)*, pp. 48-57. ACM Press, 1996
- [18] N.-Y. Lee, T. Hwang, and C.-H. Wang. "Nonrepudiable proxy signature schemes." In *Proceedings of Information Security and Privacy (ACISP98)*, LNCS 1438, pp. 415-422. Springer-Verlag, 1998.
- [19] T. Okamoto, M. Tada, and E. Okamoto. "Extended proxy signatures for smart cards". In *Proceedings of Information Security Workshop (ISW99)*, LNCS 1729, pp. 247-258. Springer-Verlag, 1999.
- [20] H.-U. Park and I.-Y. Lee. "A digital nominative proxy signature scheme for mobile communications". In *proceedings of Information and Communications Security (ICICS01)*, LNCS 2229, pp. 451-455. Springer- Verlag, 2001.
- [21] T.Malkin, S.Obana and M.Yung, "The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures", In *Proceedings of Eurocrypt 2004*, Vol. 3027 of LNCS, pp. 306-322, Springer-Verlag 2004.
- [22] H. Wang and J. Pieprzyk. "Efficient one-time proxy signatures". In *Proceedings of Asiacrypt03*, Springer-Verlag, 2003.
- [23] Liu Wen-Yuan, Tong Feng, Luo Yong-An,Zhang Feng. "A Proxy Blind Signature Scheme based on Elliptic Curve with proxy Revocation." In *Proceedings of Eighth ACIS International Conference on Software Engineering, Artificial Networking and Parallel Distributed Computing*.
- [24] K. Zhang. "Threshold proxy signature schemes". In *Proceedings of Information Security Workshop (ISW97)*, LNCS 1396, pp. 282-290, Springer-Verlag, 1997.
- [25] Zhen Chuan Chai, Zhenfu Cao."Factoring-Based Proxy Signature Schemes with Forward-Security". In *Proceedings of First International Symposium on Computational and Information Science*, LNCS 3314, pp 1034-1040, Springer Verlag.
- [26] Zuhua Shao, "Proxy Signature Schemes based on Factoring" *Information Processing letters*, Volume 85, Issue 3, 14th February 2003, Pages 137-143.
- [27] Shin-Jia Hwang, Ching-Chung Chan, "Improvement on Li et al.'s generalization of proxy signature schemes" *Computers & Security*, Volume 23, Issue 7, October 2004, Pages 615-619.
- [28] Qin Wang, Zhenfu Cao, "Identity based proxy multi-signature", *Journal of Systems and Software*, Volume 80, Issue 7, July 2007, Pages 1023-1029.
- [29] Tzer-Shyong Chen, Yu-Fang Chung, Kuo-Hsuan Huang, "A traceable proxy multisignature scheme based on the elliptic curve cryptosystem", *Applied Mathematics and Computation*, Volume 159, Issue 1, 25 November 2004, Pages 137-145.
- [30] Chien-Lung Hsu, Tzong-Sun Wu, "Efficient proxy signature schemes using self-certified public keys", *Applied Mathematics and Computation*, Volume 152, Issue 3, 13 May 2004, Pages 807-820.
- [31] Guilin Wang, Feng Bao, Jianying Zhou, Robert H. Deng, "Comments on a Threshold Proxy Signature Scheme Based on the RSA Cryptosystem", *IEEE Transactions on Knowledge and Data Engineering*, IEEE Computer Society.

A. Digital Signature Algorithm

The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS (Federal Information Processing Standard) for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS), specified in FIPS. This scheme is a digital signature scheme which is based on the difficulty of computing discrete logarithms

A. Key Generation:

- Choose a 160 bit prime q .
- Choose a L -bit prime p , such that $p = qz + 1$ for some integer z .
- Choose h , where $1 < h < p - 1$ such that $g = h^z \pmod{p} > 1$. Here g is the generator.
- Choose x where $0 < x < q$.
- Calculate $y = g^x \pmod{p}$.
- Public key is (p, q, g, y) . Private key is x .

B. Signature Generation:

- Generate a random per message value k where $0 < k < q$
- Calculate $r = (g^k \pmod{p}) \pmod{q}$
- Calculate $s = (k^{-1}(SHA(m) + x * r)) \pmod{q}$ where $SHA(m)$ is the hash function applied to the message m
- The signature is (r, s)

C. Signature Verification:

- Calculate $w = s^{-1} \pmod{q}$.
- Calculate $u1 = (SHA(m) * w) \pmod{q}$
- Calculate $u2 = r * w \pmod{q}$
- Calculate $v = ((g^{u1} * y^{u2}) \pmod{p}) \pmod{q}$
- The signature is valid if $v = r$

B. Forward Secure DSA Signature Scheme

To specify a forward-secure signature scheme, we need to (i) give a rule for updating the secret key (ii) specify the public key and (iii) specify the signing and the verification algorithms.

In saying that our forward-secure scheme is based on a basic signature scheme, we mean that, given a message and the secret key of a time period, the signing algorithm is the same as in the basic signature scheme. The public key for

Table 1: For prime p of size $|p|$ bits, $\phi^T(p)$ has a prime factor of size 160 bits.

$ p $	p	T
256	23158417847463239084 71419700173758157065 39969331281128078915 168015826259280709	56
256	23158417847463239084 71419700173758157065 39969331281128078915 168015826259280027	56
274	60708402882054033466 23318458823496583257 52137203793600391191 37804340758912662766479	77
274	60708402882054033466 23318458823496583257 52137203793600391191 37804340758912662765931	73
512	26815615859885194199 148049996411692254958731 6411847867554471228874 4352806014709395360374 8596333806855380063716 3729721017075077656238 93139892867298012168351	266

the forward-secure signature scheme is the key obtained on running T times the update rule for secret keys.

Now, we need to be able to write a verification equation relating the public key and the signature (and incorporating the time period of the signature) from which the claim of forward security can be deduced.

Here are the details.

1. Secret Key Updation

Let p be a large prime. Let $\phi(p - 1) = p_1^{r_1} \dots p_k^{r_k}$ where $p_1 < p_2 < \dots < p_k$.

Choose g such that

$$\gcd(g, p) = 1, \gcd(g, \phi(p)) = 1, \gcd(g, \phi^2(p)) = 1, \dots, \gcd(g, \phi^{T-1}(p)) = 1$$

where $\phi(p)$ is the Euler totient function and $\phi^{T-i}(p) = \phi(\phi^{T-i-1}(p))$ for $1 \leq i \leq T - 1$ with $\phi^0(p) = p$. It may be noted that a prime g chosen in the range $p_k < g < p$ satisfies the above condition. The base secret key a_0 (this is the initialisation for the secret key updation) is chosen randomly in the range $1 < a_0 < p - 1$.

The secret key a_i in any time period i is derived as a function of a_{i-1} , the secret key in the time period $i - 1$, as follows:

$$a_i = g^{a_{i-1} \pmod{\phi^{T-i+1}(p)}} \pmod{\phi^{T-i}(p)} \quad (34)$$

for $1 \leq i < T$. Once the new secret key a_i is generated for time period i , the previous secret key a_{i-1} is deleted. Thus an attacker breaking in period i will get a_i but cannot compute a_0, \dots, a_{i-1} , because of difficulty of computing discrete logarithms. For a given large prime p , though the value of $\phi^i(p)$ decreases exponentially over time i , we have determined experimentally (see Table 1) that for the following typical values of p , $\phi^i(p)$ factor into primes of size greater than 2^{160} for reasonable value of T . Therefore, we assume

that computing discrete logarithms mod $\phi^{T-i}(p)$ is hard, for $1 \leq i < T$.

2. Public Key Generation

We obtain the public key by executing the Secret Key Update Algorithm T times as follows :

$$\beta = g^{a_T} \pmod p = a_T \pmod p \quad (35)$$

3. Signature Generation:

The signature generated in any time period i is $\langle r, s, i \rangle$. The computation of r is

$$r = (g^k \pmod p) \pmod q \quad (36)$$

where k is a random number chosen such that $0 < k < p$ and $\gcd(k, (p-1)) = 1$.

The computation of s is

$$s = k^{-1}(SHA(m||i) + (\mathcal{A}(g, T-i-1, a_i) * r)) \pmod q \quad (37)$$

where SHA is a collision-resistant hash function. While hashing, i is concatenated with m to indicate the time period in which the message is signed.

The notation $\mathcal{A}(\alpha, u, v) = \alpha^{\dots^{\alpha^v}}$ we mean that there are u number of α 's in the tower and the topmost α is raised to v , i.e in the above equation there are $(T-i-1)$ number of α 's in the tower and the topmost α is raised to a_i .

Notice that the public key β can also be given in terms of a_i as,

$$\beta = \mathcal{A}(g, T-i, a_i) \pmod p, \quad (38)$$

This relation gets employed in the verification of validity of the signature.

4. Verification:

$$\begin{aligned} w &= (s)^{-1} \\ u1 &= SHA(m||i) * w \\ u2 &= r * w \\ v &= g^{u1} * \beta^{u2} \end{aligned}$$

A claimed signature $\langle r, s, i \rangle$ for the message m in time period i is accepted if

$$v = r \quad (39)$$

else rejected.

Notice that since

$$\begin{aligned} LHS &= g^{SHA(m||i)*w} \cdot (\mathcal{A}(\alpha, T-i, a_i))^{u2} \pmod p \\ &= g^{SHA(m||i)*w + \mathcal{A}(\alpha, T-i-1, a_i).rw} \\ &= g^{w(SHA(m||i) + \mathcal{A}(\alpha, T-i-1, a_i).r)} \\ &= g^{(s)^{-1}(SHA(m||i) + \mathcal{A}(g, T-i-1, a_i).r)} \\ &= g^k \\ &= r \\ &= RHS. \end{aligned}$$

a signature by an honest signer with the secret key will therefore be accepted.

Recall that the claim of security of the standard DSA signature scheme is based on the difficulty of computing discrete logarithms. The same security guarantee is obtained in the Forward-secure DSA Signature Scheme.

C. Bellare-Miner Forward-secure scheme

For the sake of completeness we describe the algorithms of the Bellare-Miner scheme.

A. Key generation:

The signer generates the keys by running the following algorithm which takes as input the security parameter k , the number l of points in the keys and the number T of time periods over which the scheme is to operate.

Pick random, distinct $k/2$ bit primes p, q each congruent to 3 mod 4. $N \leftarrow pq$. The base secret key $SK_0 = (S_{1,0}, \dots, S_{l,0}, N, 0)$ (where $S_{i,0} \xleftarrow{R} Z_N^*$ and N is a Blum-Williams integer).

For verifying signatures the verifier is given the public key PK , calculated as the value obtained on updating the base secret key $T+1$ times: $PK = (U_1, \dots, U_l, N, T)$ where

$$U_i = S_{i,0}^{2^{T+1}} \pmod N, i = 1, \dots, l.$$

B. Key evolution:

During time period j the signer signs using key SK_j . This key is generated at the start of period j by applying a key update algorithm to the key SK_{j-1} . The update algorithm squares the l points of the secret key at the previous stage to get the secret key at the next stage. Once this update is performed the signer deletes the key SK_j . Since squaring modulo N is a one way function, when the factorization of N is unknown it is computationally infeasible to recover SK_{j-1} from SK_j .

The secret key $SK_j = (S_{1,j}, \dots, S_{l,j}, N, j)$ of the time period j is obtained from the secret key $SK_{j-1} = (S_{1,j-1}, \dots, S_{l,j-1}, N, j-1)$ of the previous time period via the update rule

$$S_{i,j} = S_{i,j-1}^2 \pmod N, i = 1, \dots, l.$$

C. Signature Generation:

It has as input the secret key SK_j of the current period, the message M to be signed, and the value j of the period itself to return a signature $\langle j, (Y, Z) \rangle$ where Y, Z in Z_N^* are calculated as follows:

$$Y = R^{2^{(T+1-j)}} \pmod N \quad (40)$$

where $R \xleftarrow{R} Z_N^*$ and

$$Z = R \prod_{i=1}^l S_{i,j}^{c_i} \pmod N \quad (41)$$

with

$$c_1, \dots, c_l = H(j, Y, M) \quad (42)$$

being the l output bits of a public hash function.

D. Signature Verification:

As for verification, a claimed signature $\langle j, (Y, Z) \rangle$ for the message M in time period j is accepted if

$$Z^{2^{(T+1-j)}} = Y \prod_{i=1}^l U_i^{c_i} \pmod N \quad (43)$$

where $c_1, \dots, c_l = H(j, Y, M)$, and rejected otherwise. Notice that since

$$\begin{aligned} Z^{2^{(T+1-j)}} &= (R(\prod_{i=1}^l S_{i,j}^{c_i})^{2^{(T+1-j)}} \pmod N \\ &= Y(\prod_{i=1}^l S_{i,0}^{2^{(T+1)}c_i}) \pmod N \\ &= Y \prod_{i=1}^l U_i^{c_i} \pmod N. \end{aligned}$$

a signature by an honest signer with the secret key will be accepted.