Proxy and Threshold One-Time Signatures

Mohamed Al-Ibrahim[1] and Anton Cerny[2]

[1] Center for Advanced Computing
Computing Department
Macquarie University
Sydney , NSW 2109, Australia
ibrahim@ieee.org
[2] Department of Mathematics and Computer Science
Kuwait University
P.O. Box 5969, Safat 13060, Kuwait
cerny@mcs.kuniv.edu.kw

Abstract. One-time signatures are an important and efficient authentication utility. Various schemes already exist for the classical one-way public-key cryptography. One-time signatures have not been sufficiently explored in the literature in the branch of society-oriented cryptography. Their particular properties make them suitable, as a potential cryptographic primitive, for broadcast communication and group-based applications. In this paper, we try to contribute to filling this gap by introducing several group-based one-time signature schemes of various versions: with proxy, with trusted party, and without trusted party.

Keywords: One-time signature, proxy signature, group signatures, digital signatures, threshold cryptography, broadcast authentication

1 Introduction

One-time signatures are an important public-key cryptography primitive. They derive their importance from their fast signature verification. This is in contrast to the conventional digital signature schemes, which usually have high generation or verification computation time. One-time signatures are an ideal option for authenticating particular types of applications where receivers are of low power capability, such as smart cards, or for online applications, such as video/audio streaming, which require fast verification.

One-time signatures are efficient and secure. Typically, signature parameters are initialized well ahead of the time when messages are to be signed and verified. This allows the signer to pre-compute the signature parameters so they can be fetched by potential verifiers. Once the message is known, the signer can sign it quickly, and receivers can verify the signed message efficiently. A distinct characteristic of one-time signatures is that they are used once only. To sign a new message, the signer must initialize the signature parameters. This means that the parameters of old signatures are not reused to sign another message;

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 123–136, 2003.

otherwise, the signature parameters would be exposed. The security of one-time signatures is measured by the difficulty of forging a signature by an adversary who normally has access to a single pair: a message and its signature.

On the other hand, groups play an important role in contemporary communication. Numerous examples of group applications include the stock exchange, collaborative tasks, and many other multicast applications. Group communication has a (potentially) high communication overhead, and it is desirable that group members can authenticate their communications efficiently. Cryptographic transformations by a group of participants was the subject of investigation in so called *society-oriented cryptography* ([6], [4]). Unlike classical cryptography, society-oriented cryptography allows groups of cooperating participants to carry out cryptographic transformations. That is, society-oriented cryptography requires distribution of the power of performing a cryptographic transformation among a group of participants such that only designated subsets of the group can perform the required cryptographic operation, but unauthorized subsets cannot do so.

With the recent interest in securing group and broadcast communication, there has been a great demand for designing a new class of fast signature schemes that can handle a vast number of signatures from broadcast or group-based applications efficiently, rather than using typical signature schemes. Hence, there have been a number of attempts in society-oriented cryptography to design signature schemes for authenticating group-based scenarios.

Several schemes were proposed that use classical authentication schemes such as digital signatures (RSA [24], ElGamal [7]) for group-based transformations. However, these conventional methods typically have a high computational overhead, and hence they may not fulfill the new requirements with regard to the efficiency of the emerging applications. Besides, the nature of authenticating online real-time applications usually requires a fast authentication. That is, the extra embedded complexity which is involved in the typical digital signatures to provide extra security adds more computational time. In contrast, one-time signatures provide the required security services with less computational overhead.

As mentioned, one-time signatures are potentially far more (computationally) efficient than classical authentication methods. This leads us to explore new ways of improving the efficiency of authentication in group communication using one-time signatures. That is, we attempt to apply one-time signatures for situations where the right to execute signature operations is shared by a group of signers. We propose a scheme for a threshold proxy signature. We achieve this by introducing a few intermediate schemes. We put together the concepts of one-time signature and threshold group signature by using the idea of proxy signing. Proxy one-time signature was first used in [12] to authenticate mobile agents in low-bandwidth communications. On the other hand, and to the best of our knowledge, the problem of finding a group oriented one-time signature has not been discussed elsewhere.

In this paper, we begin by reviewing the relevant work in the area of one-time signatures. To allow application of several one-time signature schemes in a

common way, we establish a construction model for the proposed protocols. To reach our goal, we investigate the problem of one-time signature in two scenarios: with a proxy signer and with a group of signers. We start with one-time signature for the case when a signer wants to delegate his one-time signature to a proxy who would sign on his behalf. Then in Section 5, we show how it is possible to construct a threshold one-time signature using the Shamir secret sharing method. This may happen with or without the aid of a trusted party. Finally, in Section 6, we design a scheme for threshold proxy signature.

2 Related Work

Lamport [13], Rabin [16], Merkle [17] and GMR [11] are well known examples of one-time signature schemes. They share the same basic idea and are based on committing to secret keys via one-way functions. Rabin uses an interactive approach for verification of signatures with the signer. These schemes differ in their approaches, but they share the same idea: only one message can be signed using the same key. Once the signature is released, its private key is not used again; otherwise, it is possible for an adversary to compute the secret key.

A new approach to designing such signatures is the BiBa one-time signature [19]. The BiBa signature exploits the birthday paradox property. A large number of secret keys is used to find collisions among the generated keys associated with the message. This way of signing requires a long pre-computational time. Reyzin and Reyzin [23] solve BiBa's disadvantage of having a very long signing time. Their idea is to calculate the number of required keys according to the size of the message and pre-determined length of the signature. Based on this, key generation would be very fast, and hence signing is faster.

One-time signatures have been used in group communication for authenticating streaming applications in multicast communication. Gennaro and Rohatchi [9] used a chained method with one-time signature. Rohatchi used a k-times one-time signature based on on-line/off-line approach. Perrig used it in Tesla [18]. Al-Ibrahim *et al.* in [1] introduced k-sibling one-time signature for authenticating transit flows.

3 A Class of One-Time Signature Schemes

In this section, we establish a model for one-time signature schemes. The model is not aimed at introducing a new kind of signature. We want to set a common view of several well-established signature schemes in order to be able to apply any one of them in our subsequent scenarios. Not every signature scheme in our model is therefore secure and the properties of each such particular scheme are to be investigated separately.

Our model consists of a signer S and a verifier V and is described as a tuple $\mathcal{O} = (M, X, Y, h, v, \pi)$ where M is the set of messages, X, Y are finite sets, $h : X \to Y$ is a one-way hash function, $v \geq 1$ is an integer and $\pi : M \to 2^{\{1,2,\cdots,v\}}$

is a function. All parts of $\mathcal{O}$ are public. If a signer S sends a message $m \in M$ to a verifier V, the signature creation and verification proceeds as follows:

Key generation

Signer S

1. chooses v random values $s_1, s_2, \ldots, s_v \in X$ (the secret keys of the signer)
2. computes v values $p_1 = h(s_1)$, $p_2 = h(s_2), \ldots, p_v = h(s_v) \in Y$ and makes them public.

Signing

Signer S

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. sends m and the signature $(s_{j_1}, s_{j_2}, \ldots, s_{j_r})$ to the verifier V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $h_1 = h(s_{j_1}), h_2 = h(s_{j_2}), \ldots, h_r = h(s_{j_r})$
3. accepts the signature if and only if $h_1 = p_{j_1}, h_2 = p_{j_2}, \ldots, h_r = p_{j_r}$.

The model includes schemes like Lamport [13] or Merkle [17] as special cases. The schemes of Rabin [22], GMR [11], BiBa [19] are of a different type. The "better than BiBa" scheme of Reyzin and Reyzin [23], Bos and Chaum [3], and HORS++ of [21] belong to this model.

4 A Simple One-Time Proxy Signature Scheme

Delegation of rights is a common practice in the real world. A manager of an institution may delegate to one of his deputies the capability to sign on behalf of the institution while he is on holiday. For electronic transactions, a similar approach is needed to delegate the manager digital signature to the deputy.

Proxy signature is a signature scheme where an original signer delegates his/her signing capability to a proxy signer, and then the proxy signer creates a signature on behalf of the original signer. When a receiver verifies a proxy signature, he verifies both the signature itself and the original signer's delegation. Mambo, Usuda and Okamoto (MUO) in [15] established models for proxy signatures. They classified proxy signatures, based on delegation type, as full delegation, partial delegation, and delegation by warrant. In full delegation, the signer gives his secret key to the proxy. In partial delegation, the signer creates a separate secret key for the proxy, but it is derived from his secret key. In signing with warrant, the signer signs the public key. In addition, they provide various constructions for proxy signature schemes with detailed security description and analysis. Their proxy signatures provide various security services including:

- **Unforgeability**. Only the proxy signer (besides the original signer) can create a valid signature for the original signer.
- **Proxy signer's deviation**. Each valid proxy signer's signature can be detected as her signature.
- **Verifiability**. A positive verification of a proxy's signature guarantees the agreement of the original signer with this signature.
- **Distinguishability**. A valid proxy's signature can be distinguished from the original signer's signature (in polynomial time).
- **Identifiability**. The original signer can determine the identity of a proxy from his signature (if there are more proxy signers).
- **Undeniabilty.** A proxy signer cannot disavow his valid signature.

Detailed discussion may be found in [15].

Zhang in [27] noticed that the Mambo scheme does not provide

- **Nonrepudiation**. Neither the original nor the proxy signer can falsely deny later that he generated a signature.

In [27] the scheme from [15] has been enhanced to provide nonrepudiation.

The scheme in [15] allows the proxy to sign an arbitrary number of messages. Furthermore, using the warrant is not a systematic way to control the number of signed messages in electronic communication. In some situations, the signer may need to delegate his signature to the proxy for one-time/one-purpose only. For example, an autocratic manager may want, for security or administrative reasons, to restrict the proxy to signing on his behalf for one time only. Hence, a new type of proxy signature that is more restricted than the Mambo approach is needed. An efficient one-time signature can be used in this case. Kim *et al.* in [12] designed a one-time proxy signature using fail-stop signature to provide authentication to mobile agents applications. In our proposal, we use a class of one-time signatures, as those described in section 3, to design a new one-time proxy signature.

If we consider a "classical" type of scheme, where the original signer shares his secret keys with the proxy or generates new secret keys for the proxy, distinguishability and non-repudiation are not guaranteed, since both the original and the proxy signer know the secret keys. The character of a one-time signature allows us to adopt a principally new approach, where the secret keys are generated and kept by the proxy only, and the original signer has no share in the secret. The original signer only confirms the public keys and stores them with a trusted party (registry). The security properties such as unforgeability, proxy signer's deviation, verifiability, and undeniability of the scheme are the same as in the underlying one-time signature scheme $\mathcal{O}$. Introducing the proxy signer clearly does not cause any violation of these security properties, unless the signature scheme is used more than once by an unreliable proxy. Signing several messages using the same set of keys reveals too many secret keys, and an eavesdropper could easily sign a false message using them. Our suggested solution

involves the trusted party. Let us assume that, besides the public keys approved by the original signer, one more value will be stored by the proxy when signing a message. However, this action will be guarded by the trusted party and will not be allowed to be done more than once. When signing the message m, the proxy will store there the value $h(m)$. In an additional step, the verifier will compute $h(m)$ for the received message m and check it against the value from the registry. Since this value can be stored to the registry just once, only one message can be legally signed. The scheme involves a signer S, a proxy P, a verifier V, and a trusted party TP. It uses the one-time signature scheme $\mathcal{O} = (M, X, Y, h, v, \pi)$ where X is a sufficiently large set. In addition, we assume that h is extended to $h : X \cup M \to Y$ while still preserving its one-wayness.

Key generation

Signer S

1. asks P to start generating secret keys.

Proxy P

1. chooses v secret numbers : $s_1, s_2, \ldots, s_v \in X$
2. computes $p_1 = h(s_1), p_2 = h(s_2), \ldots, p_v = h(s_v)$
3. passes $(p_1, p_2, \ldots, p_v)$ to S.

Signer S

1. verifies that the p's are from P (a one-time signature of P can be used for signing the list of p's)
2. makes $(p_1, p_2, \ldots, p_v)$ public, registered to the name of S.

Signing

Proxy P

1. computes $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $q = h(m)$ and registers this value with TP as a one-time writable value
3. sends $(m, s_{j_1}, \ldots, s_{j_r})$ to V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $h_1 = h(s_{j_1}), h_2 = h(s_{j_2}), \ldots, h_r = h(s_{j_r})$
3. computes $q' = h(m)$
4. fetches $p_{j_1}, p_{j_2}, \ldots, p_{j_r}$ and q from TP
5. accepts the signature if and only if $h_1 = p_{j_1}, h_2 = p_{j_2}, \ldots, h_r = p_{j_r}$ and $q' = q$.

The proxy uses its private keys, and the public keys are stored with a trusted party; hence, the proxy cannot deny signing or revealing the secret to a third agent - a danger occurring in most of the established proxy signature schemes. Since the signer and the proxy do not share the same key, non-repudiation is achieved. Sending keys by the proxy to the signer in the key generation phase does not compromise security since these keys will become public anyway. The role of the original signer is to endorse the public-keys generated by the proxy signer to the registry. This step is crucial; otherwise, any agent may claim itself to be a proxy for the original signer.

5 A (t, n) Threshold One-Time Signature Scheme

A particularly interesting class of society-oriented cryptography which includes threshold cryptographic transformation, consists of all subsets of t or more participants from a group of n members. A digital signature is an integer issued by a signer which depends on both the signer's secret key and the message to be signed. In conventional cryptosystems, the signer is a single user. However, the process of signing may need to be shared by a group of participants. The first attempts at designing a shared signature were made by Boyd. Threshold RSA [5] and Threshold ElGamal [14] signatures are examples of threshold multisignature schemes that require the presence of t participants of the group to sign a message. Both schemes exploit the Threshold Shamir secret sharing method to generate shares of signatures.

Here, we attempt to expand the idea of threshold signatures from the conventional cryptosystems transformations into one-time signatures to benefit from its efficiency properties in speeding-up the verification process. Our model consists of a group of signers $S_i, i = 1, 2, \ldots, n$ and a verifier V. A one-time signature scheme $\mathcal{O} = (M, \mathbb{F}, Y, h, v, \pi)$ is used, where $\mathbb{F}$ is a finite field and Y is a finite set, and both are sufficiently large. A threshold value $t \leq n$ is specified in advance. Not less than t signers are required to sign a message.

5.1 With a Trusted Party

In our first scenario, two more parties are involved: a trusted distributor D, and a trusted combiner C.

The idea behind this scheme is to let the distributor D choose the secret keys of the general one-time signature scheme of the group.

The shares of these secret keys for the particular signers are computed by D using the Shamir secret sharing algorithm, and they are then distributed to the participants. For each secret key s_j, a distinct polynomial f_j of degree $t-1$ with $f_j(0) = s_j$ is used to create secret shares. A public value x_i is associated with each signer S_i; his secret share on the key s_j is then $s_{i,j} = f_j(x_i)$. The set of polynomials comprising the system is illustrated (graphically) in Figure 1. Each intersection of a polynomial with the y axis represents a secret key. Two or more shares of the same signer may be identical, since several polynomials may have

a common value in some of the points x_i (the graphs may intersect on a vertical line at x_i). This clearly does not compromise the security of the system, since this information is known only to D and the the signer. The secret keys s_j are chosen to be pairwise distinct; hence no two polynomial graphs intersect on the y axis.

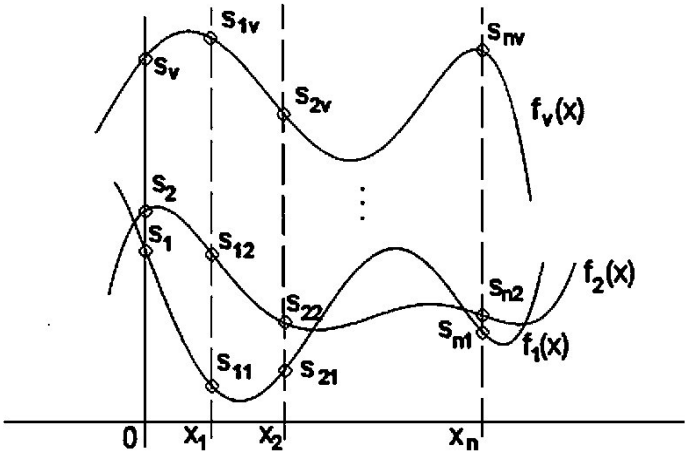

Fig. 1.

Key generation and share distribution *Distributor D*

1. chooses randomly v pairwise distinct elements $s_j \in \mathbb{F}, j = 1, \ldots, v$ (group secret keys)
2. computes the v values $p_j = h(s_j), j = 1, \ldots, v$ and makes them public (registered to the group name)
3. chooses randomly n pairwise distinct non-zero elements $x_i, i = 1, \ldots, n$ from $\mathbb{F}$ and makes them public
4. chooses randomly v polynomials $f_j(x) = f_{j,0} + f_{j,1}x + \cdots + f_{j,t-1}x^{t-1}, j = 1, \ldots, v$, satisfying $f_j(0) = f_{j,0} = s_j$
5. computes $s_{i,j} = f_j(x_i), i = 1, \ldots, n, j = 1, \ldots, v$
6. sends $(s_{i,j})_{j=1,\ldots,v}$ by a secure channel to the signer $S_i, i = 1, \ldots, n$ (secret share for the partial signer S_i)

Signing

Signer $S_i, i \in \{i_1, i_2, \ldots, i_t\}$

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. sends the partial signature $(s_{i,j_1}, s_{i,j_2}, \ldots, s_{i,j_r})$ to C.

Combiner C

1. waits to receive partial signatures from (at least) t signers $S_{i_1}, \ldots, S_{i_t}$
2. using Lagrange interpolation, recovers the polynomials $f_{j_k}(x)$, $k = 1, \ldots, r$, based on the conditions $f_{j_k}(x_{i_1}) = s_{i_1,j_k}, \ldots, f_{j_k}(x_{i_t}) = s_{i_t,j_k}$
3. finds $(s_{j_1}, s_{j_2}, \ldots, s_{j_r}) = (f_{j_1,0}, f_{j_2,0}, \ldots, f_{j_r,0})$
4. sends $(m, s_{j_1}, s_{j_2}, \ldots, s_{j_r})$ to V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. fetches p_{j_k}, $k = 1, \ldots, r$ from the registry
3. checks whether $p_{j_k} = h(s_{j_k})$, $k = 1, \ldots, r$.

Since the usual Shamir secret sharing is used, at least t signers are necessary to find any of the group secret keys. The fact that at most one message will be signed using the same signature may be guaranteed by the trusted combiner, so the multiple signature problem vanishes. In our scheme, the trusted distributor D knows all the secret keys; therefore, his reliability must be without doubt. The next version, without a trusted party, avoids such strict requirements. Observe that the combiner C knows only those secret keys which are used for the signature and which will be revealed to the verifier.

The computation of the shares involves nv times evaluation of a polynomial of degree $t-1$ by D and r times Lagrange interpolation of a polynomial of degree $t-1$ by C. In addition, D, V and each partial signer must compute $\pi(m)$ and D and V compute v and r values of the function h, respectively. The signers may compute π in parallel. It is worth noting that the verification of the one-time signature scheme is as efficient as without secret sharing.

5.2 Without a Trusted Party

The scenario without a trusted party works the same way as the one with the trusted party; the steps of the distributor D and combiner C are performed by the signers themselves. In particular, the set of n signers is used as a parallel n-processor computer.

Key generation and share distribution

Signer $S_i, i = 1, \ldots, n$

1. chooses randomly a non-zero element $x_i \in \mathbb{F}$ and makes (i, x_i) public
2. chooses randomly $s_j \in \mathbb{F}$ (secret key) for each $j = 1, \ldots, v$ such that $(j-1) \bmod n + 1 = i$
3. computes the value $p_j = h(s_j)$ and makes the pair (j, p_j) public (registered to the group name)
4. chooses randomly a polynomial $f_j(x) = f_{j,0} + f_{j,1}x + \cdots + f_{j,t-1}x^{t-1}$ satisfying $f_j(0) = f_{j,0} = s_j$
5. computes $s_{i',j} = f_j(x_{i'}), i' = 1, \ldots, n$
6. sends $s_{i',j}$ by a secure channel to the signer $S_{i'}, i' = 1, \ldots, n, i' \neq i$ (secret share for the signer $S_{i'}$)

Signing

Signer $S_i, i \in \{i_1, i_2, \ldots, i_t\}$

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. sends the partial signature $(s_{i,j_1}, s_{i,j_2}, \ldots, s_{i,j_r})$: the triple (i, j_k, s_{i,j_k}) is sent to S_{i_q} where $q = (j_k - 1) \bmod t + 1, k = 1, 2, \ldots, r$
3. using Lagrange interpolation, based on the conditions $f_{j_k}(x_{i_1}) = s_{i_1,j_k}$, $f_{j_k}(x_{i_2}) = s_{i_2,j_k}, \ldots, f_{j_k}(x_{i_t}) = s_{i_t,j_k}$,recovers the polynomial $f_{j_k}(x)$ for each complete t-tuple $((i_1, j_k, s_{i_1,j_k}), \ldots, (i_t, j_k, s_{i_1,j_k}))$ received
4. for each polynomial f_j recovered, finds $s_j = f_{j,0}$
5. for each polynomial f_j recovered, sends (m, j, s_j) to V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. fetches p_{j_k}, $k = 1, \ldots, r$ from the registry
3. waits until all triples (m, j_k, s_{j_k}), $k = 1, \ldots, r$ are received
4. checks whether $p_{j_k} = h(s_{j_k})$, $k = 1, \ldots, r$.

Let r_0 be the minimum length of a signature and v the total number of secret keys. In our scheme, each of the n signers generates $\lceil v/n \rceil$ secret keys. We claim that if $\lceil v/n \rceil (t-1) < r_0$, then at least t out of the n signers are necessary to create a valid signature. Indeed, $(t-1)$ signers know at most $\lceil v/n \rceil (t-1)$ secret keys. If the condition holds, this is not enough to create a signature of length r_0. On the other hand, the multiple signatures problem arises here again. Several messages may be signed even without malicious intention, since two independent subgroups of size t may sign two different messages. This problem may again be resolved by the "trusted registry" as in Section 4. Another solution may be an explicit synchronization of all signers before signing by one subgroup. That is, on time slot σ, all the n participants would be involved in signing the message m though there are only t actual signers. We leave open the problem of

designing a better scheme for one-time signatures that would prevent subgroups in a threshold scheme from signing more than one message at a time.

The complexity considerations from Part 5.1 are valid, except that, instead of the time necessary for computating nv polynomial values, the time for computing $\max(n, n\lceil v/n\rceil \approx v)$ values is required, since the signers may compute in parallel. In a similar way, only the time for $\lceil r/t\rceil$ Lagrange interpolations is necessary.

How realistic is the condition $\lceil v/n\rceil (t-1) < r_0$? If the scheme of Lamport ([13]) is used to sign a message of length μ, then $v = 2\mu$ are generated and the signature consists of μ keys. Our condition is satisfied if $t < n/2 + 1$.

6 A (t, n) Threshold Proxy One-Time Signature Scheme

In this section, we combine the ideas from the two previous sections and propose the following model. A group of n signers $S_i, i = 1, 2, \ldots, n$ wants to allow any subgroup of at least t signers to sign a message using a one-time signature. In our solution, the group will play the role of the original signer, who delegates his right to use a one-time signature to any subgroup of $t \leq n$ (proxy). The signature is to be verified by a verifier V. A one-time signature scheme $\mathcal{O} = (M, \mathbb{F}, Y, h, v, \pi)$ is used, where $\mathbb{F}$ is a finite field and Y is a finite set, both sufficiently large. Again, we assume that h is a one-way hash function, $h : M \cup \mathbb{F} \to Y$. The trusted party TP is required only to keep the public keys and to prevent repeated signing. The start of the keys generation should be initiated in a suitable coordinated way.

Key generation and share distribution

Signer $S_i, i = 1, \ldots, n$

1. chooses randomly a non-zero element $x_i \in \mathbb{F}$ and makes (i, x_i) public
2. chooses randomly $s_j \in \mathbb{F}$ (secret key) for each $j = 1, \ldots, v$ such that $(j - 1) \bmod n + 1 = i$
3. computes the value $p_j = h(s_j)$ and sends (j, p_j) to TP
4. chooses randomly a polynomial $f_j(x) = f_{j,0} + f_{j,1}x + \cdots + f_{j,t-1}x^{t-1}$ satisfying $f_j(0) = f_{j,0} = s_j$
5. computes $s_{i',j} = f_j(x_{i'}), i' = 1, \ldots, n$
6. sends $s_{i',j}$ by a secure channel to the signer $S_{i'}, i' = 1, \ldots, n, i' \neq i$ (secret share for the signer $S_{i'}$)

Trusted Party TP

1. verifies that each p_j is from a proper S_i (a one-time signature of S_i can be used for signing the pair (j, p_j))
2. makes $(p_1, p_2, \ldots, p_v)$ public, registered to the name of the group.

Signing

Signer $S_i, i \in \{i_1, i_2, \ldots, i_t\}$

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $q = h(m)$ and registers this value with TP; if q is different from a value already registered with TP, S_i stops signing
3. sends the partial signature $(s_{i,j_1}, s_{i,j_2}, \ldots, s_{i,j_r})$: the triple (i, j_k, s_{i,j_k}) is sent to S_{i_q} where $q = (j_k - 1) \bmod t + 1, k = 1, 2, \ldots, r$
4. using Lagrange interpolation, based on the conditions $f_{j_k}(x_{i_1}) = s_{i_1,j_k}$, $f_{j_k}(x_{i_2}) = s_{i_2,j_k}, \ldots, f_{j_k}(x_{i_t}) = s_{i_t,j_k}$, recovers the polynomial $f_{j_k}(x)$ for each complete t-tuple $((i_1, j_k, s_{i_1,j_k}), \ldots, (i_t, j_k, s_{i_1,j_k}))$ received.
5. for each polynomial f_j recovered, finds $s_j = f_{j,0}$
6. for each polynomial f_j recovered, sends (m, j, s_j) to V.

Verification

Verifier V

1. after receiving the first triple (m, j_k, s_{j_k}) finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $q' = h(m)$
3. fetches p_{j_k}, $k = 1, \ldots, r$ and q from TP
4. waits until all triples (m, j_k, s_{j_k}), $k = 1, \ldots, r$ are received
5. checks whether $p_{j_k} = h(s_{j_k})$, $k = 1, \ldots, r$ and $q' = q$.

As in Part 5.2, each of the signers knows at most $\lceil v/n \rceil$ secret keys of the group. Therefore, $(t-1)$ signers will not be enough to sign a message only if $\lceil v/n \rceil (t-1) < r_0$, where r_0 is the minimum signature length for messages under consideration.

The computation of the shares involves the time for $\max(n, n\lceil v/n \rceil \approx v)$ evaluations of a polynomial of degree $t-1$ and the same number of applications of h, since the signers may compute in parallel. Similarly, only the time for $\lceil r/t \rceil$ Lagrange interpolations of a polynomial of degree $t-1$ is required. In addition, V and each partial signer must compute $\pi(m)$ and V must compute r values of the function h.

6.1 Special Case: $t = 1$

A particular case of interest in this scheme is when $t = 1$, which depicts the anycast model. The anycast authentication problem was discussed in [2] and a solution was proposed based on a conventional digital signature. Briefly, the anycast model represents the situation where any of a group of n servers (signers) may provide the same (equivalent) service to a client (verifier). The method of nominating the actual signer is an optimization problem, and it is done by the network infrastructure based on a number of criteria such as less communication overhead, more available memory, and others. In the solution, an additional party - a group coordinator - may behave as the original signer in our $(1, n)$-threshold scheme while the servers behave as the proxies. The original signer delegates his

power to n proxy signers and the verifier verifies a message from "one" of the proxy signers. Although the above $(1, n)$- threshold scheme of one-time signature is theoretically correct, it is not of practical concern since the signer needs to generate different secret keys for each proxy correspondence.

7 Conclusion and Future Work

In this paper, we have proposed several schemes related to authentication with one-time signature. The first case deals with the implementation of a one-time signature in proxy delegation; the second shows how to use a (t, n) threshold one-time signature in a group of signers, and the third scheme combines the above two ideas into a (t, n) threshold proxy one-time signature. An extension to this work, left as an open problem, is to design a one-time signature scheme to prevent multiple message signing using the same set of one-time signature keys.

References

1. M. Al-Ibrahim and J. Pieprzyk, "Authentication of transit flows and K-siblings one-time signature," in *Advanced Communications and Multimedia Security*, B. Jerman-Blazic and T. Klobucar, ed., pp. 41–55, Kluwer Academic Publisher, CMS'02, Portoroz – Slovenia, September 2002.
2. M. Al-Ibrahim and A. Cerny, "Authentication of anycast communication," in the proc. of Second MMM-ACNS'03, St-Petersburg, Russia, LNCS 2776, pp. 425–429, Springer-Verlag, 2003.
3. J.N.E. Bos, D.Chaum, "Provably unforgeable signatures," *Advances in Cryptology – CRYPTO '92*, Ernest F. Brickell ed., LNCS 740, pp.1–14, Springer-Verlag, 1992.
4. C. Boyd, "Digital Multisignatures," in *Cryptography and coding* (H. Beker and F. Piper, eds.), pp. 241–246, Clarendon Press, 1989.
5. Y. Desmet and Y. Frankel. "Threshold cryptosystems," in G. Brassard, editor, *Advances in Cryptology – Crypto'89*, LNCS 435, pp. 307–315, Springer-Verlag, 1990.
6. Y. Desmedt, "Society and group oriented cryptography: a new concept," in *Advances in Cryptology – Proceedings of CRYPTO '87*, C. Pomerance, ed., LNCS 293, pp. 120–127, Springer-Verlag, 1988.
7. T. ElGamal, "A Public key cryptosystem and a signature scheme based on discrete Logarithms, " *IEEE Trans. on Inform. Theory*, vol. IT-31, pp. 469–472, July 1985.
8. S. Even, O Goldreich, S. Micali. "On-line/Off-line digital signatures," *Journal of Cryptology*, volume 9, number 1, pp. 35–67, 1996.
9. R. Gennaro and P. Rohatchi, "How to sign digital streams," *Advances in Cryptology – CRYPTO'97*, LNCS 1249, pp. 180–197, Springer-Verlag, 1997.
10. O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, 33(4):pp. 792–807, October 1986.
11. S. Goldwasser, S. Micali, and C. Rackoff, "A Digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal on Computing, 17, pp. 281–308, 1988.

12. H. Kim, J. Baek, B. Lee, and K. Kim, "Secret computation with secrets for mobile agent using one-time proxy Signature," Proc. of *SCIS'2001*, pp. 845–850, IEEE press, 2001.
13. L. Lamport, "Constructing digital signatures from a one-way function," Technical report CSL-98, SRI International, Palo Alto, 1979.
14. C.M. Li, T. Hwang, and N.Y. Lee, "Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders," in A. De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, LNCS 950, pp. 194–204, Springer-Verlag, 1995.
15. M. Mambo, K. Usuda, E. Okamoto, "Proxy signatures for delegating signing operation," IEICE Trans. Fundamentals, vol. E79-A, no.9, pp.1338–1354, 1996.
16. A. Menezes, P. Van Oorschot, and S. Vanstone. "Handbook of Applied Cryptography," *CRC Press*, Boca Raton, 1997.
17. R. Merkle. "A Certified digital signature," *Advances in Cryptology – CRYPTO'89,* LNCS 435, pp. 218–238, Springer-Verlag, 1989.
18. A. Perrig, R. Canetti, J.D. Tygar, D. Song, "Efficient Authentication and signing of multicast streams over lossy channels," *IEEE Symposium on Security and Privacy*, pp. 56–73, 2000.
19. A. Perrig. "The BiBa one-time signature and broadcast authentication protocol," *ACM, CCS'01*, pp. 28–37, 2001.
20. J. Pieprzyk, T. Hordajano, and J. Seberry. "Fundamentals of computer security," Springer-Verlag, 2003.
21. J. Pieprzyk, H. Wang, C. Xing. "Multiple-time signature schemes secure against adaptive chosen message attack," *SAC'03* (10th workshop on Selected Areas of Cryptography), LNCS, Springer-Verlag, 2003, to appear.
22. M.O. Rabin, "Digitalized signatures," R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, Foundations of Secure Computation, pp. 155–168, Academic Press, 1978.
23. L. Reyzin and Natan Reyzin, "Better than BiBa: short one-time signatures with fast signing and verifying," *ACISP'02*, LNCS 2384, pp. 144–152, Springer-Verlag, 2002.
24. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.
25. P. Rohatchi. "A Compact and fast hybrid signature scheme for multicast packet authentication," in *Proc. of 6th ACM Conference on Computer and Communications Security*, pp. 93–100, 1999.
26. A. Shamir. "How to share a secret," *Communications of the ACM*, 22:612–613, 1979.
27. K. Zhang, "Threshold proxy signature scheme," In proc. of *ISW'97,* LNCS 1396, pp. 282–290, Springer-Verlag, 1997.