Efficient One-time Proxy Signatures

Huaxiong Wang and Josef Pieprzyk

Centre for Advanced Computing – Algorithms and Cryptography
Department of Computing
Macquarie University
Sydney, NSW 2109, AUSTRALIA
hwang/josef@ics.mq.edu.au

Abstract. One-time proxy signatures are one-time signatures for which a primary signer can delegate his or her signing capability to a proxy signer. In this work we propose two one-time proxy signature schemes with different security properties. Unlike other existing one-time proxy signatures that are constructed from public key cryptography, our proposed schemes are based one-way functions without trapdoors and so they inherit the communication and computation efficiency from the traditional one-time signatures. Although from a verifier point of view, signatures generated by the proxy are indistinguishable from those created by the primary signer, a trusted authority can be equipped with an algorithm that allows the authority to settle disputes between the signers. In our constructions, we use a combination of one-time signatures, oblivious transfer protocols and certain combinatorial objects. We characterise these new combinatorial objects and present constructions for them.

1 Introduction

In general, digital signatures can be divided into two classes. The first class includes one-time signatures and their variants based on one-way functions without trapdoors. These schemes can be used to sign a predetermined number of messages only, we will call them *one/multiple-time signature schemes* (examples of such schemes includes one-time signatures by Lamport [16] and Rabin [27], but also multiple-time signatures by Rohatgi [32], by Reyzin and Reyzin [30], and by Pieprzyk, Wang and Xing [26]). The second class of schemes is based on public-key cryptography and they can be used to sign an unlimited number of messages. The RSA [29] and the ElGamal [10] signatures represent this class.

One-time signatures were first proposed by Rabin [27] and Lamport [16] and are based on the idea of committing public keys to secret keys using one-way functions. For more than 25 years, various variants of Rabin's schemes have been proposed and investigated by many researchers (see, for example, [3, 4, 11, 16, 20]). Indeed, one-time signatures have found many interesting applications [7, 21], including on-line/off-line signatures [9], digital signatures with forward security properties [1], broadcast authentication protocols [25] and stream-oriented authentication [32] etc.

One of the main advantages of one-time signatures is their reliance on one-way functions without trapdoors that can be implemented using fast hash functions such as SHA-1 or MD5. The resulting signatures are the order of magnitude faster than signatures based on public cryptography. With the advent of low-powered, resource-constrained, small devices, such as cell phones, pagers, Palm pilots, smart cards etc. in recent years, one-time signatures have attracted more and more attention, as an attractive alternative to the traditional signatures based on public key cryptography (see, for example [15, 25, 30]).

Although digital signatures have been successfully applied to ensure the integrity, authenticity, and non-repudiation for the electronic documents, standard signatures (both based on public-key cryptography and on one-way functions) alone are too inflexible and inefficient to handle many practical requirements in new applications. Thus, many variants of the standard signatures with additional functionalities have been proposed. These include blind, undeniable, and group signatures to mention a few. Motivated by applications that require the power to sign to be transfered from one person to another, Mambo *et al* [19] proposed proxy signatures. Proxy signatures allow a designated person, called a *proxy*, to sign on behalf of a primary signer. A proxy signature convinces a verifier that the primary signer has delegated the signing power to the proxy and that the proxy has signed the message.

To our best knowledge, all the previously published proxy signatures are based on public-key cryptography. Most of the proxy signatures can be viewed as modifications of the ElGamal signature and their security typically relies on the assumption of the difficulty of the discrete logarithm problem (the DL assumption). In addition, these proxy schemes can generally be used for signing multiple messages and for multiple proxy signers.

In this paper, we will study *one-time* proxy signatures (or simply OTP signatures). As the name suggests, we consider one-time signatures with the additional *proxy* functionality. It should be noted that the notion of one-time proxy signature itself is not new, and it has been proposed by Kim *et al* [15] in a different context. Their signature is a variant of the ElGamal signature (or more precisely, a variant of one-time fail-stop signature [13]) and its security rests on the DL assumption. The motivation behind their work is to limit the power of the proxy signer so the proxy signer can sign once only. In contrast, our motivation is to enable the primary signer to delegate a proxy to sign in the applications where one-time signatures (based on one-way functions) are used.

To define our proxy signatures, we employ two basic cryptographic primitives as the building blocks. The first one is a one-time (or multiple-time) signature primitive based on one-way functions. The second building block is an oblivious transfer (OT) primitive. We then combine these primitives with certain combinatorial objects to obtain our OTP signatures. We formulate the general framework for proxy signatures, define their security goals and attacks against them. We then show that the efficiency of any OTP signature can be measured by the properties of the underlying combinatorial objects. We introduce *proxy patterns* that characterise the properties of these OTP signatures. Next, we give constructions for the desired proxy patterns, using polynomials over finite fields and error-correcting codes, and link them with other combinatorial structures (such as Steiner systems).

The rest of the paper is organised as follows. In Section 2, we introduce our model of one-time proxy signatures. In Section 3, we consider candidates for the two building blocks that can be used to construct one-time proxy signatures. In Section 4, we propose a simple scheme for one-time proxy signatures and later we describe a basic scheme and analyse its security. In Section 5, we analyse the basic scheme and and its security against the swallow attacks. Finally, Section 6 concludes the paper.

2 The Model

A *proxy signature* enables the primary signer to delegate his/her signing capability to a proxy signer so the proxy signer can generate a signature on behalf of the primary signer. Mambo *et al* [19] introduced the concept of proxy signature. They defined three classes of delegation: *full delegation, partial delegation* and *delegation by warrant*. A full delegation scheme assumes that the primary signer and the proxy signer have the same secret key, so the proxy signer can sign any message that is indistinguishable from the signature generated by the primary signer. A signature with partial delegation allows the primary signer to delegate the power of signing to a proxy in such a way that the signatures generated by the primary and proxy signers are different. This is normally done by making verification algorithms different for primary and proxy signatures. In other words, proxy signatures are distinguishable from primary signatures. A signature with delegation by warrant requires an additional piece of message (called a warrant) that determines the proxy signer that is delegated by the primary signer. Signatures with full delegation do not provide non-repudiation while signatures with partial delegation do. Signatures with delegation by warrant can be implemented using double signatures and therefore, they are not as efficient as signatures with full or partial delegations.

In this paper, we are interested in one-time signatures that allow full delegation with an added feature that allows to trace the authorship of the signature (if both proxy and primary signers agree to settle a dispute). Being more precise, we are going to consider proxy signatures with full delegation, in which the private signing key of the proxy signer is derived from the private key of the primary signer. In particular, we restrict our attention to signatures that can be used once only.

Informally, a *one-time proxy signature scheme* (OTP signature) includes two parties: a *primary signer* and a *proxy signer* together with the following three algorithms.

Key Generation: For a given security parameter, it outputs a pair of private and public keys for the primary signer and a private key for the proxy signer. The key generation may involve a two-party protocol run between the primary and proxy signers, or a multi-party protocol that is run amongst three parties: the primary signer, the proxy, and a trusted authority.

Singing: For an input that consists of a message to be signed and the private key of the signer (either primary or proxy), it outputs a valid signature.

Verifying: For an input that includes a pair (a message and a signature) and the public key of the primary signer, it outputs either *accept* or *reject*.

In the following, we consider the basic security requirements imposed on OTP signatures. If an OTP signature satisfies the requirements, it is called *secure*.

Unforgeability: It is infeasible for any third party (that has not been involved in signing) to forge a message/signature that passes the signature verification. This means that if a signature has been generated by the primary signer, no body (including the proxy) can forge a message/signature. Also if the signature has been generated by the proxy, then no body (including the primary signer) can forge a message/signature.

Verifiability: For a valid signature, a verifier is convinced that the primary signer has agreed to sign a message (either the primary signer has signed it or the proxy has).

Traceability: In case of a dispute between the primary and proxy signers, there exists a tracing algorithm that reveals the identity of the actual signer. That is, the algorithm guarantees that it should be infeasible for

- the primary singer to sign a message and to claim later that it has been signed by the proxy signer.
- the proxy signer to sign a message and to claim later that it has been signed by the primary signer.

We note that the model of our OTP signature is slightly different from previous proxy signatures in the sense that there is only one public key of the primary signer for the signature verification. Thus, from a verifier point of view, signatures generated by primary or proxy signers are indistinguishable (like in the full delegation). However, the tracing algorithm guarantees the non-repudiation property for the primary signer and the proxy singer. Thus, unlike in full delegation signatures, the primary singer and the proxy signer have different private keys for signature generation, and in case a dispute occurs between the two potential signers, the tracing algorithm is called to resolve it. We argue that the indistinguishable between the signatures by the primary signer and the proxy signer is an interesting property, for example, it can be used to protect the privacy of the actual signer. However, in this paper we are not going to explore it beyond this point.

3 Building Blocks

In this section, we review two cryptographic primitives that are needed in the our constructions of proxy signatures.

3.1 One-time Signature

One-time signatures are based on one-way functions. Rabin published the first one-time signature based on a private-key encryption or a one-way function without a trapdoor [27], requiring interaction between the signer and the verifier. Lamport [16] gave a non-interactive one-time signature using a one-way function. The idea of Lamport is as follows. For a given one-way function f, one selects two random strings x_0, x_1 as the secret key, and publishes $f(x_0)$ and $f(x_1)$ as the public key. Then the single-bit message $b \in \{0, 1\}$ can be signed by revealing x_b. Various modifications of the Lamport signature with improved efficiency and functionalities have been proposed (see, for example [2, 4, 5, 9, 12, 14, 21, 25, 30, 32]).

As our building block, we are going to use a one-time signature defined as follows. Let b, t, k be integers such that $\binom{t}{k} \geq 2^b$. Let T denote the set $\{1, 2, \ldots, t\}$ and $\mathcal{T}_k$ be the family of k-subsets of T. Let S be a one-to-one mapping from $\{0, 1, \ldots, 2^b - 1\}$ to $\mathcal{T}_k$ such that for a message m, $S(m)$ assigns a unique k-element subset from $\mathcal{T}_k$. Let f be a one-way function operating on ℓ-bit strings (ℓ is a security parameter).

The signature scheme consists of three algorithms: *key generation*, *signing* and *verification*. For a given security parameter ℓ, the key generator chooses at random t strings s_i of the length ℓ bits and creates the secret key $SK = (s_1, \ldots, s_t)$. The public key is the image of the secret key obtained using the one-way function f, i.e., $PK = (v_1, \ldots, v_t)$ such that $v_1 = f(s_1), \ldots, v_t = f(s_t)$.

To sign a b-bit message m, the signer interprets m as an integer between 0 and $2^b - 1$ and computes $S(m) = \{i_1, \ldots, i_k\} \in \mathcal{T}_k$. The value $s_{i_1}, \ldots, s_{i_k}$ is the signature of m.

To verify a signature $(s'_1, s'_2, \ldots, s'_k)$ on a message m, the verifier again interprets m as an integer between 0 and $2^b - 1$ and computes $\{i_1, \ldots, i_k\}$ as the m-th k-element subset of $\mathcal{T}_k$. Finally, the verifier checks whether $f(s'_1) = v_{i_1}, \ldots, f(s'_t) = v_{i_t}$.

Definition 1. *We call the above one-time signature scheme a (t, k) one-time signature scheme and denote it by $\mathcal{O} = (T, S, f)$, or simply by $\mathcal{O}$. The parameters (t, k) specify efficiency of the signature.*

Note that the Bos-Chaum one-time signature scheme [2] is a special case of the (t, k) scheme in which $k = t/2$. Note also that for a (t, k) one-time signature $\mathcal{O} = (T, S, f)$, the most expensive part of computation is the implementation of the mapping S. In [30], Reyzin and Reyzin present two algorithms for implementation for S with computation costs of $O(tk \log^2 t)$ or $O(k^2 \log t \log k)$. In [26], Pieprzyk *et al* give more efficient implementations for S through the explicit constructions of S using polynomials over finite fields, error-correcting codes, and algebraic curves.

3.2 Oblivious Transfer (OT)

An oblivious transfer (OT) refers to a two-party protocol executed between a sender S and a receiver R. The goal of the protocol is to transfer the knowledge about an input string held by the sender to the receiver in such a way that the receiver learns some part of the input but the sender cannot figure out which part of the input is now known to the receiver. Consider a 1-out-n oblivious transfer (OT_1^n) protocol. The sender S has n secrets (strings) $m_1, m_2, \ldots, m_n$, and is willing to disclose one of them (m_α) to R for some index α chosen by R. However, R does not want to reveal its choice of the index α to S and at the same time, S does not want R to gain any information about other secrets $m_i, i \neq \alpha$. In general, we may have a k-out-n oblivious transfer (OT_k^n), in which R may choose k indices out of n.

The concept of oblivious transfer has been introduced by Rabin in 1981 [28] and it has been extensively studied (see, for example, [8, 22, 23]). Here is an example of OT_1^n proposed recently by Tzeng [33], which is among the most efficient OT protocols proposed so far. Let g and h be two (public) generators in a q-order group G_q, where q is prime. Assume that the secret input of S is $m_1, m_2, \ldots, m_n \in G_q$, and the choice of R is α, $1 \leq \alpha \leq n$. The protocol proceeds as follows.

1. $R \to S : y = g^r h^\alpha$ for a random $r \in \mathcal{Z}_q$,
2. S randomly chooses n elements $k_i \in \mathcal{Z}_q$ and

$$S \to R : c_i = (g^{k_i}, m_i(y/h^i)^{k_i}),\ 1 \leq i \leq n.$$

3. R computes $m_\alpha = b/a^r$, assuming $c_\alpha = (a, b)$.

It is proved in [33] that in the above OT_1^n protocol, the confidentiality of the receiver choice is unconditionally secure and the confidentiality of un-chosen secrets is at least as strong as the hardness of the decision Diffie-Hellman problem. As to computations required in the protocol, the receiver needs to compute 2 modular exponentiations and the sender computes $2n$ modular exponentiations.

4 One-time Proxy Signatures

Our basic idea behind the constructions of OTP signatures is as follows. The primary signer generates n private/public key pairs for one time signatures, say $(sk_1, pk_1), \ldots, (sk_n, pk_n)$. The proxy signer gains one of the n private keys, say sk_i in such a way that the primary signer does not know, which key was obtained by the proxy signer, i.e., the primary signer does not know the index i. The primary signer publishes the public key $pk_1, \ldots, pk_n$ in an authenticated way. The proxy signer uses sk_i to sign the message, which can be verified by anyone who knows the public key. Note that the verification of signatures generated by primary and proxy signers is the same.

To prevent cheating by signers, a tracing algorithm has to be carefully designed. The algorithm should be run by a trusted authority and should identify the true signer with a high probability. Note that the oblivious transfer enables us to identify the true signer. To do this, the trusted authority always asks the proxy to sign the disputed message again. If the proxy is unable to produce a different signature it means that either the proxy really signed the message or the primary signer has applied the same secret key as proxy (this event happens with the probability $1/n$).

4.1 A simple proxy signature scheme

We present a simple and somewhat trivial scheme to illustrate the basic idea. Then we improve its efficiency using some combinatorial techniques. The scheme is based on a (t, k) one-time signature $\mathcal{O} = (T, S, f)$ and an oblivious transfer protocol OT_1^n (or OT_k^n), and it works as follows.

Key Generation: It consists of the following three steps.

- The primary signer randomly chooses an $n \times t$ array $A = (s_{ij})_{n \times t}$ as her private key. Each row holds t secret keys of an instance of the (t, k) one-time signature $\mathcal{O}$. The public key is $V = (v_{ij})_{n \times t}$, where $v_{ij} = f(s_{ij})$ and f is the one-way function from $\mathcal{O}$.
- The primary and proxy signers execute an OT_1^n (or OT_k^n) protocol. At the end of the protocol, the proxy signer learns one row from A, say $(s_{i1}, \ldots, s_{it})$, as his private key, but nothing more. The primary signer has no information about the index i.
- The proxy singer applies f to $(s_{i1}, \ldots, s_{it})$ and compares the results with the ith row of public array V. If the check fails to hold, the proxy exits the scheme and complains to the primary signer.

Signing: The proxy signer applies the ith row of A, i.e., $(s_{i1}, \ldots, s_{it})$, as his private key of the one-time signature $\mathcal{O}$ and signs the message m. That is the proxy signer first computes $S(m) = \{j_1, \ldots, j_k\} \subseteq \{1, \ldots, t\}$ and then reveals m and the signature $\delta = \{(s_{ij_1}, \ldots, s_{ij_k}), i\}$.

Verifying: This part follows the steps necessary to verify an instance of the (t, k) one-time signature.

Security We discuss the security requirements of the scheme. Obviously, unforgeability and verifiability of the OTP signature follow directly from the unforgeability and verifiability of the underlying one-time signature $\mathcal{O}$. What we need to consider is the traceability of the true signer (in case of cheating attempts from either the proxy or the primary signer).

Unforgeability against the primary signer: Assume that the primary singer wants to cheat. She generates a signature for a message m and later claims that it was generated by the proxy signer. Note that to sign m, the primary signer has to choose a row of A and to sign using the chosen instance of one-time signature. Suppose that she has chosen jth row of A. The generated signature is $\delta_j = \{(s_{ji_1}, \ldots, s_{ji_k}), j\}$, where $S(m) = \{i_1, \ldots, i_k\}$. The proxy signer can prove that the signature was not generated by him, by revealing another signature for m using his private key $(s_{i1}, \ldots, s_{it})$. That is, he reveals the signature $\delta_i = \{(s_{ii_1}, \ldots, s_{ii_k}), i\}$, which shows that $\delta_i \neq \delta_j$. As the proxy signer knows only one row of the private keys, he can only sign the message with one of the rows, so δ_j must have been generated by the primary signer. The OT protocol provides unconditional security for the proxy signer and the probability of success of the primary signer is $1/n$.

Unforgeability against the proxy singer: Suppose that the proxy signer wants to cheat, he generates a signature, later denies it and claims that the primary signer (or someone else) has generated the signature. His claim can be accepted only if he can generate a different signature for the same message. In other words, the proxy is able to produce two different signatures for the same message. This is impossible unless, he is able to break the OT protocol or to invert the one-way function.

We stress that the tracing algorithm is called only if the dispute between the primary signer and the proxy signer occurs. The knowledge of a valid signature alone is not sufficient to identify the actual signer (the signature provides full delegation).

Efficiency We look at the efficiency of the scheme. The signing and verification of the signature are exactly the same as the underlying one-time signature scheme, so could be very fast. The key generation requires n times costs of key generation for one-time signatures, plus the cost of running an OT_1^n (or OT_t^n) protocol. The length of public and secret keys increases n times as well. However, observe that the key generation, which is the most expensive part of computations, can be precomputed. Furthermore, an expensive OT protocol can be avoided if a third trusted party helps during the key generation. The private key of the primary signer can be discarded after the key generation. In the next section we propose methods to reduce the public key length.

4.2 The basic proxy signature scheme

To decrease the probability of successful cheating by the primary signer, it is required to increase the parameter n and consequently the number of rows in A. This causes that the simple proxy signature secure against a dishonest primary signer must have a long private/public key. We show that the simple proxy signatures can be converted into proxy signatures with shorter public keys using combinatorial techniques.

Definition 2. *Given a set $X = \{x_1, \ldots, x_M\}$ and an $n \times t$ array $C = [c_{ij}]$ with entries from X. The array C is called a (t, k, n, M)* proxy pattern*, denoted by $PP(t, k, n, M)$, for a (t, k) one-time signature if*

1. *each row of C contains t different elements of X,*
2. *any two distinct rows of C have at most $k-1$ common elements, i.e., for any $i \neq j$,*

$$|\{c_{i1}, \ldots, c_{it}\} \cap \{c_{j1}, \ldots, c_{jt}\}| < k.$$

For a given $PP(t, k, n, M)$, we combine it with a (t, k) one-time signature to construct an OTP signature that is a generalisation of the simple scheme presented above. Without the loss of generality, assume that $C = (c_{ij})$ is a $PP(t, k, n, M)$ with entries taken from $X = \{1, \ldots, M\}$ and $\mathcal{O} = (T, S, f)$ is a (t, k) one-time signature. Our basic proxy signature works as follows.

Key Generation: It goes through the following three steps.

- The primary signer randomly chooses M distinct values $(s_1, s_2, \ldots, s_M)$ as the private key (for example, each s_i is an ℓ-bit string if the underlying one-time signature $\mathcal{O}$ is defined for the security parameter ℓ). The public key is $V = (v_1, \ldots, v_M)$, where $v_i = f(s_i), i = 1, \ldots, M$.
- The primary and proxy signers execute an OT_t^M protocol. At the end of the protocol, the proxy signer learns the ith row of C, that is $(s_{c_{i1}}, \ldots, s_{c_{it}})$, as his private key, but nothing more. The primary signer has no information about the index i.
- The proxy singer applies f to $(s_{c_{i1}}, \ldots, s_{c_{it}})$ and checks the results with the corresponding components of the public key V. If the check fails, the proxy aborts and complains.

Signing: For a given message m, the proxy signer applies his private key $(s_{c_{i1}}, \ldots, s_{c_{it}})$ to the one-time signature $\mathcal{O}$ and signs the message. That is, the proxy signer first computes $S(m) = \{j_1, \ldots, j_k\} \subseteq \{1, \ldots, t\}$ and then reveals the signature $\delta = \{(s_{c_{ij_1}}, \ldots, s_{c_{ij_k}}), i\}$.

Verifying: It follows the verification of the (t, k) one-time signature (applied to the appropriate instance of the one-time signature) in a straightforward manner.

It is easy to see that the security of this scheme is similar to the security of the simple scheme. The traceability is guaranteed by the properties of the proxy pattern C, that is, any two rows will have at most $k-1$ common elements. Since a signature requires the knowledge of k secret values of the private key, the proxy signer can resolve disputes by showing two valid signatures (corresponding to two different rows of C).

The main advantage of the basic signature scheme is a reduction of the length of public key (and the corresponding private key) from nt values to M values. In the remainder of this section, we will give constructions for proxy patterns with small M and derive a bound on the minimal value for M.

4.3 Constructions of proxy patterns

It is easy to see that the simple signature uses a trivial $PP(t, k, n, nt)$ for any $k, 1 \leq k \leq t$. By fixing k, as this is the case for the underlying (k, t) one-time signature, we are able to construct a $PP(t, k, n, M)$ such that M is significantly smaller than nt, and so to reduce the length of the public key.

Assume $GF(q)$ is a finite field with q elements and $a_1, \ldots, a_t$ are t distinct elements from $GF(q)$. We construct a $PP(t, k, n, M)$ as follows. Consider a set $X = \{a_1, \ldots, a_t\} \times GF(q)$ and all polynomials of the degree at most $k - 1$ over $GF(q)$. Next write them as $g_1(x), \ldots, g_{q^k}(x)$. Note that there are q^k such polynomials. Further define a $q^k \times t$ array $C = [c_{ij}]$ with entries taken from X, so

$$c_{ij} = (a_j, g_i(a_j)), \quad \text{for } i = 1, 2, \ldots, q^k, j = 1, 2, \ldots, t.$$

Now we show that C is a $PP(t, k, q^k, qt)$. Indeed, for $1 \leq i \leq q^k$, the ith row of C is

$$((a_1, g_i(a_1)), (a_2, g_i(a_2)), \ldots, (a_t, g_i(a_t))).$$

Thus, for $i \neq j$,

$$\begin{aligned}
& |\{((a_1, g_i(a_1)), \ldots, (a_t, g_i(a_t))\} \cap \{((a_1, g_j(a_1)), \ldots, (a_t, g_j(a_t))\}| \\
&= |\{a \mid g_i(a) = g_j(a)\}| \\
&= |\{a \mid (g_i - g_j)(a) = 0\}| \\
&< k
\end{aligned}$$

otherwise there are k or more than k roots for the polynomial $g_i - g_j$. But $g_i - g_j$ is a polynomial of degree at most k, it follows that $g_i = g_j$ which contradicts that $i \neq j$. We have proved the following result.

Theorem 1. *Let q be a prime power. For any integers t, k such that $k \leq t \leq q$, there exists a $PP(t, k, q^k, qt)$.*

Note that for the simple proxy signature, a $PP(t, k, q^k, q^k t)$ is required. Thus, for the fixed parameters t, k and q^{k+1}, we can reduce the number of elements in the public key from $q^k t$ for the simple proxy signature to qt in the basic proxy signature.

A generalisation of the above polynomial construction uses error-correcting codes. Let Y be an alphabet of q elements. An (N, W, D, q) *code* is a set $\mathcal{M}$ of W vectors in Y^N such that the Hamming distance between any two distinct vectors in $\mathcal{M}$ is at least D. Consider an (N, W, D, q) code $\mathcal{M}$. We write each codeword as $m_i = (m_{i1}, \ldots, m_{iN})$ with $m_{ij} \in Y$, where $1 \leq i \leq W, 1 \leq j \leq N$. For a set $X = \{1, \ldots, N\} \times Y$, we define a proxy pattern $C = (c_{ij})$ as follows,

$$c_{ij} = (j, m_{ij}), \quad \text{for } i = 1, 2, \ldots, W, j = 1, 2, \ldots, N.$$

Now for each distinct i, j, we have

$$\begin{aligned}
& |\{c_{i1}, c_{i2}, \ldots, c_{iN}\} \cap \{c_{j1}, c_{j2}, \ldots, c_{jN}\}| \\
&= |\{(k, m_{ik}) : 1 \leq k \leq N\} \cap \{(k, m_{jk}) : 1 \leq k \leq N\}| \\
&= |\{k : m_{ik} = m_{jk}\}| \\
&< N - D + 1.
\end{aligned}$$

This shows that the array C constructed above is a $PP(N, N - D + 1, W, Nq)$. We then have

Theorem 2. *If there exists an (N, W, D, q) code, then there exists a $PP(N, N - D + 1, W, Nq)$.*

In the coding theory, it is known that for given k and q there are constructions (e.g. using algebraic geometry codes [24]) for (N, W, D, q) codes for which $N = O(\log W)$. In the context of proxy patterns, this means that there exists $PP(N, N - D, W, Nq)$ in which $N = O(\log W)$. Applying this observation to one-time proxy signature, we can reduce the number of elements in the public key from $O(n)$, for the simple proxy signature, to $O(\log n)$ for the proxy signature based on the coding construction.

4.4 Bounds for proxy patterns

To minimise the success probability of cheating by the primary signer, we need to have a $PP(t,k,n,M)$ for which the value n is as large as possible while other parameters t,k and M are fixed. In the following we derive an upper bound for such n.

Theorem 3. *For any $PP(t,k,n,M)$, the following inequality holds*

$$n \leq \frac{\binom{M}{k}}{\binom{t}{k}}.$$

Proof. Assume that $C=[c_{ij}]$ is a $PP(t,k,n,M)$ with entries taken from an M-set of X. For each row i, we associate a subset B_i of X, i.e., $B_i=\{c_{i1},\ldots,c_{it}\}\subseteq X$, where $i=1,\ldots,n$. Clearly, $|B_i|=t$ and $|B_i\cap B_j|<k$ for all i,j where $i\neq j$. For each $1\leq i\leq n$, denote $\mathcal{R}_i$ to be the family of all the k-subsets of B_i. This implies that $|\mathcal{R}_i|=\binom{t}{k}$. Now we claim that $\mathcal{R}_i\cap\mathcal{R}_j=\emptyset$ for each $i\neq j$. If this claim is not true or $B\in\mathcal{R}_i\cap\mathcal{R}_j$ is a k-subset of X, then B is a k-subset of both B_i and B_j, which contradicts the fact that $|B_i\cap B_j|<k$. Thus we have

$$\binom{M}{k}\geq|\cup_{i=1}^{n}\mathcal{R}_i|=n|\mathcal{R}_i|=n\binom{t}{k}.$$

The desired result follows immediately. □

Next, we show that the bound in Theorem 3 is tight for some parameter set. Recall that a *Steiner system* $S(k,t,M)$ is a pair $(X,\mathcal{B})$, where X is a set of M elements called *points* and $\mathcal{B}$ is a family of t-subsets of X called *blocks*, such that every k-subset of points is contained in a unique block. It is known that the number of blocks of an $S(k,t,M)$ is $\binom{M}{k}/\binom{t}{k}$.

Corollary 1. *An $PP(t,k,n,M)$ with $n=\binom{M}{k}/\binom{t}{k}$ exists if and only if there exists an $S(k,t,M)$.*

Proof. Let $(X,\mathcal{B})$ be an $S(k,t,M)$. For each block, associate a row of an $n\times t$ array in a natural way, i.e., entries of the ith row are assigned to the elements in the block B_i. It is easy to see that assignment gives rise to a $PP(t,k,n,M)$ with $n=\binom{M}{k}/\binom{t}{k}$.

On the other hand, assume that C is a $PP(t,k,\binom{M}{k}/\binom{t}{k},M)$ with entries from M-set X, each row of C is a subset of X, we obtain a set system $(X,\mathcal{B})$ where $\mathcal{B}=\{B_i : 1\leq i\leq\binom{M}{k}/\binom{t}{k}\}$. It is clear that each k-subset of X appears in at most one block. So we need to show that it is contained in at least one block. Using the same notation as in Theorem 3, we know that each block B_i contributes $\binom{t}{k}$ k-subsets $\mathcal{R}_i$ of X. Since $\mathcal{R}_i$s are disjoint and there are $\binom{M}{k}/\binom{t}{k}$ such $\mathcal{R}_i$, which gives rise all the $\binom{\mathcal{M}}{k}$ possible choices of k-subsets of X, that means that any k-subset must be in one of the R_i. This concludes the proof. □

5 Proxy Signatures Secure against Swallow Attacks

Consider the following attack: suppose the primary signer has seen a valid signature (m, δ) produced by the proxy. She knows that the private key of the proxy signer is the ith row of the proxy pattern. Now the primary signer *swallows* the signature generated by the proxy signer, and generates the signature for another new message, using the private key of the proxy signer. In this case, the proxy signer is unable to prove his innocence. We will call it, the *swallow attack*.

In order to protect proxy signatures against the swallow attack, the primary signer should not be able to guess the private key of the proxy from a signature produced by the proxy. Looking at a message and its signature, the primary signer should not be able to determine the private key of the proxy. In other words, a single proxy signature should point at many (potential) private keys of the proxy. On the other hand, there should not be *too many* private keys corresponding to a given proxy signature. Otherwise, the proxy signature can be subject to an attack in which the primary signer chooses at random the proxy private key (without looking at the signature) and succeeds with a high probability. Based on this observation, we propose a new proxy signature that is secure against the swallow attack.

First we need some notation. Let $C = (c_{ij})$ be an $n \times t$ array with entries from an M-set of X. For any $1 \leq i \leq n$ and $1 \leq j_1 \leq j_2 \leq \cdots \leq j_k \leq t$, we denote

$$C[i; j_1, j_2, \ldots, j_k] = \{\ell \mid c_{\ell j_1} = c_{i j_1}, \ldots, c_{\ell j_k} = c_{i j_k}\}.$$

In other words, $C[i; j_1, j_2, \ldots, j_k]$ is the set of indices of the rows which are identical to ith row when restricted to the $j_1, \ldots, j_k$ columns.

Definition 3. *Given a set $X = \{x_1, \ldots, x_M\}$. An $n \times t$ array $C = (c_{ij})$, with entries from X, is called a (λ_1, λ_2)-strong (t, k, n, M) proxy pattern, denoted by (λ_1, λ_2)-SPP(t, k, n, M) for a (t, k) one-time signature if*

1. *each row of C contains t different elements of X,*
2. *any two distinct rows of C have at most k common elements, i.e., for any $i \neq j$,*
$$|\{c_{i1}, \ldots, c_{it}\} \cap \{c_{j1}, \ldots, c_{jt}\}| \leq k.$$
3. *for any row $1 \leq i \leq n$ and any k columns $1 \leq j_1 \leq j_2 \leq \cdots \leq j_k \leq t$,*
$$\lambda_1 \leq |C[i; j_1, j_2, \ldots, j_k]| \leq \lambda_2.$$

We now combine a (λ_1, λ_2)-SPP(t, k, n, M) and a (t, k) one-time signature to construct an OTP signature secure against the swallow attack. Assume $C = (c_{ij})$ is a (λ_1, λ_2)-SPP(t, k, n, M) with entries taken from $X = \{1, \ldots, M\}$ and $\mathcal{O} = (T, S, f)$ is a (t, k) one-time signature. The signature works as follows.

Key Generation: It consists of the following three steps.

- The primary signer randomly chooses M distinct elements $(s_1, s_2, \ldots, s_M)$ as the private key (for example, each s_i is a ℓ-bit string if the private key of underlying one-time signature $\mathcal{O}$ consists of ℓ-bits strings). The public key is $V = (v_1, \ldots, v_M)$, where $v_i = f(s_i), i = 1, \ldots, M$.

- The primary and proxy signers execute an OT_t^M protocol. At the end of the protocol, the proxy signer learns a t-subset of X that is the ith row of C, i.e., $(s_{c_{i1}}, \ldots, s_{c_{it}})$, as his private key, but nothing more. The primary signer has no information about the index i.
- The proxy singer applies f to $(s_{c_{i1}}, \ldots, s_{c_{it}})$ and checks the results by comparing them to the corresponding components of the public key V. If the check fails, the proxy aborts and complains.

Signing: To sign a message m, the proxy signer computes $S(m) = \{j_1, j_2, \ldots, j_k\}$ and $C[i; j_1, \ldots, j_k]$. Then he randomly chooses $\ell \in C[i; j_1, \ldots, j_k]$, and reveals $\delta = \{(s_{c_{\ell j_1}}, \ldots, s_{c_{\ell j_k}}), \ell\}$ as the signature.

Verifying: It follows the verification of the (t, k) one-time signature (applying to the ℓth row) in a straightforward manner.

Clearly, the unforgeability against the third party is the same as the underlying one-time signature scheme $\mathcal{O}$. Next we show that the scheme is secure against regular attacks and the swallow attacks from the primary signer.

Lemma 1. *The probability that the primary signer succeeds in the regular attack (without seeing any signature) is at most* λ_2/n.

Proof. In this attack, the primary signer generates a signature and later claims that it is generated by the proxy signer. She succeeds if the proxy signer fails to prove that he has not generated the signature. As the primary signer has no information about the index i chosen by the proxy signer, she may try to guess it. Assume that she has chosen the index j. For a message m, the primary signer computes $S(m) = \{j_1, \ldots, j_k\}$ and reveals the signature $\{s_{c_{jj_1}}, \ldots, s_{c_{jj_k}}, \ell\}$, where $\ell \in \mathcal{C}[j; j_1, \ldots, j_k]$. Note that if $j \notin C[i; j_1, \ldots, j_k]$, then the proxy can sign the message m using a different key from the ith row, which results in different signature of the primary signer. The primary signer succeeds if and only if $j \in C[i; j_1, j_2, \ldots, j_k]$. Since C is a (λ_1, λ_2)-SPP$(t, k+1, n, M)$, we know that $|C[i; j_1, \ldots, j_k]| \leq \lambda_2$ and the result follows. $\square$

Lemma 2. *The probability that the primary signer succeeds in the swallow attack (having seen a signature) is at most* $\max\{1/\lambda_1, \lambda_2/n\}$.

Proof. In this attack, the primary signer has seen a message/signature pair (m, δ) generated by the proxy signer. Next she swallows the data and generates another message/signature pair (m', δ'). She succeeds if the proxy signer fails to prove that there is a cheating from the primary signer. Suppose that the proxy signer has chosen the index i. For a signature (m, δ) generated by the proxy signer, we may assume that $\delta = \{(s_{c_{\ell j_1}}, \ldots, s_{c_{\ell j_k}}), \ell\}$, where $S(m) = \{j_1, \ldots, j_k\}$ and $\ell \in C[i; j_1, \ldots, j_k]$. Having seen the signature δ, the primary signer knows that the secret index chosen by the proxy signer is one of the elements in $C[\ell; j_1, \ldots, j_k]$. One attack strategy from the primary signer is to randomly choose $j \in C[\ell; j_1, \ldots, j_k]$ and use secret key from jth row to generate the signature (m', δ'). She succeeds with probability $1/|C[\ell; j_1, \ldots, j_k]|$ that $j = i$. If $j \neq i$, then the proxy signer can generate the signature for m', say δ''. It can be seen that $\delta' \neq \delta''$, which means that the proxy can create two signatures for the same message m' using two different row keys. This proves that the primary signer attempted to cheat. Another strategy for the primary signer is to choose $j \notin C[\ell; j_1, \dots, j_k]$. In this case, she succeeds if and only if $j \in C[i; j'_1, \dots, j'_k]$, where $S(m') = \{j'_1, \dots, j'_k\}$. As in the proof of Lemma 1, the probability of a successful attack using this strategy is at most λ_2/n. Therefore, the overall success probability of the attack is bounded by $\max\{1/\lambda_1, \lambda_2/n\}$. □

Previously, we have used polynomials over a finite field to construct a $PP(t, k, q^k, qt)$. We will show that this construction can be extended for (q, q)-SPP$(t, k-1, q^k, qt)$.

Theorem 4. *The polynomial construction for a $PP(t, k, q^k, qt)$ given in Section 4 results in a (q, q)-SPP$(t, k-1, q^k, qt)$.*

Proof. We already know that the polynomial construction gives rise to a $PP(t, k, q^k, qt)$, $C = (c_{ij})$. To show that C is a (q, q)-SPP$(t, k-1, q^k, qt)$. We need to show that for any $1 \leq i \leq q^k$ and $1 \leq j_1 \leq j_2 \dots, j_{k-1} \leq t$, we have

$$C[i; j_1, j_2, \dots, j_{k-1}] = q.$$

In other words, we need to show that for any $k-1$ distinct elements $a_{j_1}, \dots, a_{j_{k-1}} \in GF(q)$, and any $k-1$ elements $\alpha_1, \dots, \alpha_{k-1} \in GF(q)$, there are exactly q polynomials g of degree at most $k-1$ such that

$$g(a_{i_1}) = \alpha_1, \dots, g(a_{i_{k-1}}) = \alpha_{k-1}. \tag{1}$$

Indeed, choose $a \in GF(q) \setminus \{a_{i_1}, \dots, a_{i_{k-1}}\}$, then a polynomial g satisfying (1) is uniquely determined by the value of $g(a)$, there are q different possible choices for $g(a)$ which in turn give rise to q possible polynomial polynomials satisfying (1). This proves our desired result. □

It should be noted that constructions for strong proxy patterns can also be based on error-correcting codes. The argument follows the one developed in Section 4.3. However, it is not clear how the parameters λ_1, λ_2 are related to the parameters of the codes. We believe that it is an interesting problem for further research.

6 Conclusions

In this work, we have studied one-time proxy signature schemes. Unlike other existing one-time proxy signature scheme that are constructed using public-key cryptography, we have proposed one-time proxy signatures based on one-way functions. These signatures preserve the basic functionalities and properties of one-time signatures (including their fast generation and verification) but also allow the primary signer to delegate the power of signing to a chosen proxy.

The one-time proxy signatures permit full delegation for which potential verifiers are not able to distinguish primary signers from proxy. However, in case of a dispute between the signers about the authorship of a signature, a trusted authority is able to run an algorithm to resolve the dispute. The algorithm asks the proxy to re-generate a signature for the disputed message. If the proxy is able to produce a signature different from the disputed one, then the true signer of the signature is the primary signer. Otherwise, the proxy has generated the signature.

One-time proxy signatures can be especially useful where there is a need for fast generation and verification together with a need to share power of signing. Applications may include authentication of streams of packets in a distributed environment with mirror servers generating proxy signatures.

Our approach is based on a combination of certain type of existing one-time signature with some combinatorial objects. While the former can be optimised using the known techniques in the literature, the latter are new combinatorial objects we introduce in this paper and so are of independent interest. In particular, the structures of strong proxy patterns are far from clear, and providing efficient constructions for them is an interesting research problem.

Acknowledgement

The work was in part supported by Australian Research Council Discovery grants DP0345366 and DP0344444.

References

1. M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme, *Advances in Cryptology – Asiacrypt'00*, LNCS, **1976**(2000), 116-129.
2. J. N. E. Bos and D. Chaum. Provably unforgeable signature, *Advances in Cryptology – Crypto'92*, LNCS, **740**(1993), 1-14.
3. M. Bellare and S. Micali. How to sign given any trapdoor function. *Journal of Cryptology,* **39**(1992), 214-233.
4. D. Bleichenbacher and U. Maurer. Directed acyclic graphs, one-way functions and digital signatures, *Advances in Cryptology – Crypto'94*, LNCS, **839**(1994), 75-82.
5. D. Bleichenbacher and U. Maurer. On the efficiency of one-time digital signatures, *Advances in Cryptology – Asiacrypt'96*, LNCS, **1163**(1996), 145-158.
6. D. Bleichenbacher and U. Maurer. Optimal tree-based one-time digital signature schemes, *STACS'96*, LNCS, **1046**(1996), 363-374.
7. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications, *Advances in Cryptology – Crypto'94*, LNCS, **839**(1994), 234-246.
8. G. Di Crescenzo, T. Malkin and R. Ostrovsky. Single database private information retrieval implies oblivious transfer, *Advances in Cryptology - Eurocrypt'00*, LNCS, 2000, 122-138.
9. S. Even, O. Goldreich and S. Micali. On-line/off-line digital signatures, *Journal of Cryptology*, **9**(1996), 35-67.
10. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory.* **31**(1985), 469-472.
11. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, **17**(1988), 281-308.

12. A. Hevia and D. Micciancio. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. *Advances in Cryptology – Asiacrypt'02*, LNCS, **2501**(2002), 379-396.
13. T. P. Pedersen and B. Pfitzmann. Fail-stop signatures. *SIAM Journal on Computing*, **26/2**(1997), 291–330.
14. Y.-C Hu, A. Perrig and D.B. Johnson. Packet Leashes: A defense against wormhole attacks in wireless Ad Hoc Networks. *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, 2003, to appear.
15. H. Kim, J. Baek, B. Lee and K. Kim. Secret Computation with secrets for mobile agent using one-time proxy signature. The 2001 Symposium on Cryptography and Information Security, Oiso, Japan.
16. L. Lamport. Constructing digital signatures from a one way function. *Technical Report CSL-98*, SRI International, 1979.
17. L. Lamport. Password authentication with insecure communication. *Communication of the ACM*, **24**(11), 1981, 770-772.
18. B. Lee, H. Kim and K. Kim. Strong proxy signature and its applications. The 2001 Symposium on Cryptography and Information Security, Oiso, Japan.
19. M. Mambo, K. Usuda and E. Okamoto. Proxy signatures: Delegation of the power to sign messages. *IEICE Trans. Fundamentals*, Vol. E79-A (1996), 1338-1353.
20. R.C. Merkle. A digital signature based on a conventional function. *Advances in Cryptology – Crypto'87*, LNCS, **293**(1987), 369-378.
21. R.C. Merkle. A certified digital signature. *Advances in Cryptology – Crypto'87*, LNCS, **435**(1990), 218-238.
22. M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. *Proceedings of the 31st ACM Symposium on Theory of Computing*, 1999, 245-254
23. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. SODA01, 2001.
24. H. Niederreiter and C. P. Xing, *Rational Points on Curves over Finite Fields: Theory and Applications,* Cambridge University Press, LMS 285, 2001.
25. A. Perrig. The BiBa one-time signature and broadcast authentication. *Eighth ACM Conference on Computer and Communication Security*, ACM, 2001, 28-37.
26. J. Pieprzyk, H. Wang and C. Xing. Multiple-time signature schemes secure against adaptive chosen message attacks. *the 10th annual workshop on Selected Areas in Cryptography (SAC03)*, LNCS, to appear.
27. M.O. Rabin. Digitalized signatures. *Foundations of Secure Communication*, Academic Press, 1978, 155-168.
28. M.O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard University, 1981.
29. R.L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, **21**(1978), 120-12.
30. L. Reyzin and N. Reyzin. Better than BiBa: Short one -time signatures with fast signing and verifying. *Information Security and Privacy (ACISP02)*, LNCS, **2384**(2002), 144-153.
31. R. Rivest and A. Shamir. PayWord and MicroMint: two simple micro payment schemes. *Tech. Rep.,* MIT Lab. for Computer Science, 1996.
32. P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. *6th ACM conference on Computer and Communication Security*, 1999, 93-100.
33. W-G Tzeng. Efficient 1-out-n Oblivious Transfer Schemes. PKC'02, LNCS, 159-171.