JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 17, 879-898 (2001)

An Efficient Construction for Fail-Stop Signature for Long Messages*

REI SAFAVI-NAINI, WILLY SUSILO AND HUAXIONG WANG
Centre for Computer Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
E-mail: {rei, wsusilo, huaxiong}@uow.edu.au

The security of ordinary digital signature schemes relies on a computational assumption. Fail-stop signature (FSS) schemes provide security for a signer against a forger with unlimited computational power by enabling the signer to provide a proof of forgery, if it occurs. Signing long messages using FSS requires a hash function with provable security which results in slow signature generation. In this paper we propose a new construction for FSS schemes based on linear authentication codes which does not require a hash function, and results in a much faster signature generation at the cost of slower verification, and a longer secret key and signature. An important advantage of the scheme is that the proof of forgery is the same as a traditional FSS and does not rely on the properties of the hash function. The scheme can be used in a distributed setting where signature generation requires collaboration of k signers. The paper concludes with some open problems.

Keywords: fail-stop signature schemes, authentication codes, linearised polynomials, one-time signature, threshold signature

1. INTRODUCTION

Security of an *ordinary digital signature* relies on a computational assumption, that is that there is no efficient algorithm to solve a particular problem. This means that if an enemy can solve the underlying hard problem, he can successfully forge a signature, and there is no way for the signer to prove that a forgery has occurred. To provide protection against an enemy with unlimited computational power who can always solve the underlying hard problem, fail-stop signature (FSS) schemes have been proposed [16, 27]. Loosely speaking, an FSS is a signature scheme augmented by a proof system which allows the signer to prove that a forged signature was not generated by him/her. To achieve this property, the signature scheme has many secret keys that correspond to the same public key and the sender uses a specific one of them. An unbounded enemy who has solved the underlying hard problem and knows the set of all secret keys cannot determine which secret key is actually used by the sender. In the case of a forgery, that is signing a message with a randomly chosen secret key, the sender can use his secret key

Received January 30, 2001; accepted July 10, 2001.
Communicated by Chi Sung Laih.
* The preliminary version of this paper has appeared in Indocrypt 2000.
* This work is in part supported by Australian Research Council Grant Number A49703076.

to generate a second signature for the same message which will be different with overwhelming probability from the forged one. The two signatures on the same message can be used as a proof that the underlying computational assumption is broken and the system must be stopped - hence the name *fail-stop*, and thus, FSS schemes' provide unconditional security for the signer. However security for the receiver is computational and relies on the difficulty of the underlying hard problem. FSS schemes in their basic form are one time primitives, and so the key can be used for signing a single message.

FSS schemes and their variants have been studied by numerous authors (see, for example, [20, 21, 23-25]).

Signing long messages A commonly used method for signing an arbitrarily long message using a traditional signature scheme is by the *hash-then-sign* method. Using this method, the message is first hashed and then the signature scheme is applied to the hash value. In FSS a similar method can be used. However, as noted in [16], the proof of forgery will no longer be based on showing that the underlying assumption of the signature scheme is broken; rather, it will be by showing that a collision for the collision-resistant hash function used for hashing is found. This implies that to have an acceptable proof of forgery, a hash function which is based on a computational assumption must be used. In [3, 5] hash functions based on the discrete logarithm and factorization assumption are constructed, and it is shown that they require on average one multiplication for each bit of the message, and the size of the hash value is equal to the size of the modulus. The results is that for long messages FSS schemes have a slow signature generation process (for example one million multiplications for a one megabyte file).

An alternative approach is to have an FSS scheme that can be directly used for arbitrarily long messages. An efficiency measure [24] for FSS is *information rate*, ρ, which is the ratio of the message length to the signature length. Using FSS for arbitrarily long messages and without using hash functions means that FSS with high value of ρ must be designed. The highest value of ρ for the known schemes is 1 and is achieved by a scheme proposed in [24]. For other schemes, $\rho \leq \frac{1}{2}$ and so none of the known schemes can be efficiently used for direct signing of long messages.

Other efficiency measures of FSS schemes are the lengths of the secret key, public key and the signature [16]. The most efficient FSS with respect to the first three parameters is a discrete logarithm-based system due to van Heijst and Pedersen [25] (or *vHP scheme*). The scheme in [21] has the same efficiency as vHP.

In this paper we propose a new FSS scheme for which ρ can be chosen arbitrarily low and so can be used for direct signing of long messages. This means that no hash function is required, and so security of the resulting FSS is the same as a traditional FSS construction (based on the difficulty of DL in this case). The proposed scheme requires much less computation for signing a long message compared to the hash-then-sign method and using provably secure hash functions [3, 5]. More specifically, the signing process in our scheme is around $K/2$ times faster than the vHP scheme (using a known provable secure hash functions), where K is the length of the message in the vHP scheme (at least 151 bits [15]). The drawback of the scheme compared to the vHP is that the signature verification process is slower. It also requires larger sizes for the secret key, the public key and the signature. Table 3 compares various parameters of the two schemes. Faster signing process makes the scheme attractive in environments where

the client, such as a smart card or mobile phone has limited computational power but the host is a powerful computer. The construction follows a general approach for constructing FSS scheme from unconditionally secure authentication code (A-code) proposed in [20] and uses a special class of A-codes, called linear A-codes, in which the set of encoding rules written as vectors over F_q form a vector space. These A-code are of independent interest because of their linearity. We give the construction of a linear A-codes using linearised polynomials over finite fields that can be used to sign arbitrary length messages.

An attractive feature of using linear A-codes to construct FSS schemes is their flexibility. That is, because of the algebraic properties of the underlying A-codes such FSS schemes can be easily extended into a distributed setting. We show how to extend our construction to the threshold FSS.

Previous Works The first construction of fail-stop signature [27] is a one-time signature scheme (similar to [14]), and results in bit by bit signing of a message, which is very impractical. In [17] an efficient single-recipient FSS to protect clients in an on-line payment system, is proposed. The main disadvantage of this system is that signature generation is a 3-round protocol between the signer and the recipient and so it has high communicational cost. The size of the signature is twice the length of the message. In [25], an efficient FSS that uses the difficulty of the discrete logarithm problem as the underlying assumption is presented. This is the most efficient scheme with respect to the first three parameters and results in a signature which is twice the size of the message. For the rest of this paper, we refer to this scheme as *vHP scheme*. In [24], another scheme which is nearly as efficient as the vHP scheme is proposed.

In [16, 18], a formal definition of FSS schemes is given and a general construction using *bundling homomorphism* is proposed. The construction has provable security, and all existing FSS with provable security are instances of this construction. It can be proved that for a system with security level δ for the signer, the signature length and the length of secret key required for signing a single message are at least $2\delta - 1$ and $2(\delta - 1)$, respectively.

In [23], an RSA-based FSS scheme is proposed. The construction follows the vHP scheme but is less efficient and produces signatures that are four times the length of the original message.

In [20], a general construction of FSS schemes from authentication codes is proposed. It is shown that a scheme that fits into the general construction of [16] can also be obtained by using this construction. However, it is not known if the two general constructions are equivalent.

The paper is organised as follows. In section 2, we briefly review the basic definitions and properties of FSS schemes, A-codes and the paradigm of general construction of FSS from A-codes. In section 3, we introduce the notion of linear A-codes and present a construction method of FSS from linear A-codes. In section 4 we give an efficient construction of linear A-code from linearised polynomials over finite fields. We provide some efficiency analysis of our new construction by comparing with the previously existing FSS and show that the new schemes are particularly efficient for signing long messages in section 6. We then generalise our basic FSS to (k, k) threshold FSS and discuss its extension to (k, n) threshold FSS in section 6. We conclude the paper in section 7.

2. PRELIMINARIES

2.1 Fail-Stop Signature Schemes

Similar to an ordinary digital signature scheme, an FSS scheme consists of three phases:

1. *Key generation*: The signer and the centre through a two-party protocol generate a pair of *secret key*, *sk*, and *public key*, *pk*, and make *pk* public. This is different from ordinary signature schemes where key generation can be performed by the signer individually, without the involvement of other parties.
2. *Sign*: For a message *m*, the signer uses the signature algorithm *sign* to generate the signature $y = sign(sk, m)$, and sends the pair (m, y) to the receiver(s).
3. *Test*: For a message-signature pair, the receiver(s) uses the public key *pk* and a *test* algorithm test the acceptability of the signature.

It also includes two more polynomial time algorithms:

4. *Proof*: An algorithm for proving a forgery.
5. *Proof-test*: An algorithm for verifying that the proof of forgery is valid.

A secure FSS scheme must satisfy the following additional properties [16, 18, 26].

1. If the signer signs a message, the recipient must be able to verify the signature (*correctness*).
2. A polynomially bounded forger cannot create forged signatures that successfully pass the verification test (*recipient's security*).
3. When a forger with unlimited computational power succeeds in forging a signature that passes the verification test, the presumed signer can construct a proof of forgery and convinces a third party that a forgery has occurred (*signer's security*).
4. A polynomially bounded signer cannot create a signature that he can later prove to be a forgery (*non-repudiability*).

To achieve the above properties, for each public key there exists many matching secret keys such that different secret keys create different signatures on the same message. The real signer knows only one of the secret keys, and can construct one of the many possible signatures. An enemy with unlimited computing power, although he can generate all the signatures, he does not know which one will be generated by the true signer. Thus, it will be possible for the signer to provide a proof of forgery by generating a second signature on the message with a forged signature, and use the two signatures to show the underlying computational assumption of the system is broken, hence proving the forgery.

An FSS in its basic form is a *one-time signature scheme* that can only be used for signing a single message. However, it is possible to extend an FSS scheme to be used for signing multiple messages [1, 3, 25].

Security of an FSS is broken if 1) a signer can construct a signature and later provide a proof that is forged; or 2) an unbounded forger succeeds in constructing a signature that the signer cannot prove that it is forged. These two types of forgeries are independent and so two different security parameters, ℓ and δ, are used to show the level of security against the two types of attacks. More specifically, ℓ is the security level of the recipient against forgery of the signer, and δ is that of the signer against the unbounded forger. It is proved [16] that a secure FSS is secure against adaptive chosen plain-text attack, and for all $c > 0$ and large enough ℓ, probability of success of a polynomially bounded forger is bounded by ℓ^{-c}. For an FSS with security level δ for the signer, the probability of success of an unbounded forger is limited by $2^{-\delta}$. In this case we simply call the scheme (ℓ, δ)-*secure*.

2.2 Authentication Codes

In the conventional model of unconditionally secure authentication systems, there are three participants: a *transmitter*, a *receiver* and an *opponent*. The transmitter wants to send a message to the receiver using a public channel which is subject to active attack. The opponent can impersonate the sender by inserting a message into the channel, or substitute a transmitted message with another message. To protect against these attacks, transmitter and receiver use an authentication code (A-code).

A *systematic* A-code (or A-code *without secrecy*) is an A-code where the codeword (message) generated for a source state (information to be authenticated) is obtained by concatenating an authenticator (or a tag) to the source state. The code can be specified by a triple (S, $\mathcal{T}$, $\mathcal{E}$) of finite sets together with a (authentication) mapping f: $S \times \mathcal{E} \to \mathcal{T}$. Here S is the set of source states, $\mathcal{E}$ is the set of keys and $\mathcal{T}$ is the set of authenticators. To send a source state $s \in S$ to the receiver, the transmitter uses his secret key $e \in \mathcal{E}$ that is shared by the receiver to construct a message $m = (s, t)$ where $t = f(s, e) \in \mathcal{T}$. When the receiver receives the message $m = (s, t)$, she uses her secret key to check the authenticity by verifying if $t \stackrel{?}{=} f(s, e)$. If equality holds, the message m is *valid*.

An opponent may insert a message $m' = (s', t')$ into the channel without observing any previous communication, or substitute a message $m = (s, t)$ sent over the channel with another message $m' = (s', t')$. The two attacks are called *impersonation* and *substitution*, respectively. A message (s, t) is *valid* if there exists a key e such that $t = f(s, e)$. We assume that there is a probability distribution on the source states which is known to all the participants. Given this distribution, the receiver and the transmitter will choose a probability distribution for $\mathcal{E}$. We will denote the probability of success of the opponent in impersonation and substitution by P_I and P_S, respectively. Let $P(\cdot)$ and $P(\cdot|\cdot)$ denote the probability distribution of the message space $S \times \mathcal{T}$. Then we have

$$P_I = \max_{s,t} P((s, t) \text{ valid}) \text{ and}$$

$$P_S = \max_{s,t} \max_{s \neq s', t'} P((s', t') \text{ valid} \mid (s, t) \text{ observed}).$$

If we further assume that the keys and the source states are uniformly distributed, then the deception probabilities can be expressed as

$$P_I = \max_{s,t} \frac{|\{e \in \varepsilon : t = f(s,e)\}|}{|\varepsilon|},$$

$$P_S = \max_{s,t} \max_{s \neq s',t'} \frac{|\{e \in \varepsilon : f(s,e), t' = f(s',e)\}|}{|\{e \in \varepsilon : t = f(s,e)\}|}.$$

Since the opponent can choose between the two attacks, the overall deception probability of an A-code is defined as $P_D = \max\{P_I, P_S\}$. An A-code is ϵ-*secure* if $P_D \leq \epsilon$. One of the fundamental results in the theory of A-codes is the *square root bound* [9], which states that $P_D \geq 1/\sqrt{|\varepsilon|}$, and the equality holds only if $|S| \leq \sqrt{|\varepsilon|}+1$. The square root bound gives a direct relation between the key size and the protection that we can expect to obtain.

2.3 A General Construction of FSS From A-codes

A general construction of FSS from A-codes is given in [20]. The construction is for a single-message and uses two families of A-codes, $\mathcal{A} = \{(S_K, \mathcal{T}_K, \mathcal{E}_K) : K \in N\}$ and $\mathcal{A}' = \{(S_K, \mathcal{T}_K, \mathcal{E}'_K) : K \in N\}$, a family of polynomial time collision intractable bundling hash functions $H = \{h_K : K \in N\}$ where $h_K : E_K \mapsto E'_K$, and a family of polynomial time collision intractable hash functions $H' = \{h'_K : K \in N\}$, where $h'_K : T_K \mapsto T'_K$ and the property that for any choice of key K, and for an arbitrary $e \in E_K$ the following is satisfied for all $s \in S_K$:

If $e(s) = t$, and $h_K(e) = e'$, then $e'(s) = t'$ and $h'_K(t) = t'$

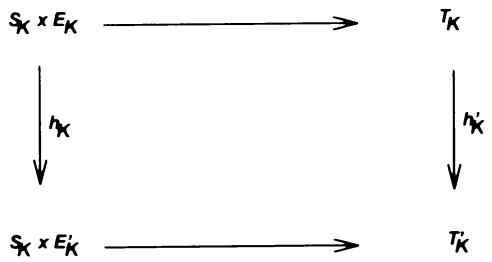


Fig. 1. General construction of FSS from A-code.

That is, we have the communicative diagram as shown in Fig. 1 and satisfying

$$h'_K(f(s_K, e_K)) = f'_K(s_K, h_k(e_K)),\ \forall s_K \in S_K,\ e_K \in \mathcal{E}_K$$

W can construct an FSS as follows:

Index K is the pre-key and is determined by a pre-key generation algorithm gen(k, k', τ) which takes the following parameters as input: (i) τ, the bundling degree of the hash function, (ii) k, the difficulty of finding collision for h, and (iii) k' which is the difficulty of finding collision for h'. Once K is determined, $(S_K, \mathcal{T}_K, \mathcal{E}_K)$, $(S_K, \mathcal{T}_K, \mathcal{E}'_K)$, h_K and h'_K are fixed, and the following stages are defined:

- *Main key generation*: The signer chooses $e_K \in \mathcal{E}_K$ as his secret key (encoding function) and constructs $e'_K = h_K(e_K)$ as his public key (verification function).

- *Signing*: The signature for the message $s \in S_K$ is given by $t = e_K(s)$.
- *Testing of the signature*: A signature t on a message s is verified if $h'_K(t) = e'_K(s)$.
- *Proof of forgery*: Given an acceptable signature t_1 on s where $t_1 \neq e_K(s)$, the signer produces $t = e_K(s)$ as the proof of forgery.

It is proven in [20] that the above construction results in the following theorems:

Theorem 2.1 The above construction has the following properties:

1. Correct signatures pass the test.
2. A polynomially bounded signer cannot construct a signature and a valid proof of forgery.
3. If t' is an acceptable signature on s' and $t' \neq e_K(s')$, the signer obtains a valid proof of forgery.

Theorem 2.2 Let ℓ and δ denote security parameters of the scheme described above. Then for all pairs of secret and public keys, e, e', and given a message and signature pair (s, t) where $t = e(s)$, the probability of a signature t' on a message s' that satisfies $e'(s') = t'$ also satisfies $e(s) = t$ for an enemy with unlimited power, is at most $|V| / |W|$, where

$$V = \max_{s',t'} \{e: e \in E(e', (s, t)), e'(s') = h'(t'), e(s') = t'\}$$

and $W = E(e', (s, t)) = \{ e': e'(s) = t \}$

3. CONSTRUCTION OF FSS FROM LINEAR A-CODES

In this section, we introduce a new class of A-codes, called *linear A-codes*, and present a construction of FSS schemes by combining linear A-codes and a one-way function $f_{p,g}$ based on the discrete logarithm.

In the rest of the paper, p is a prime and F_p is the finite field of order p (we may regard F_p as Z_p). We also use $V(n, q)$ to denote an n-dimensional vector space over a finite field with order q.

Let $(S, \mathcal{T}, \mathcal{E})$ be an authentication code with the authentication mapping f: $S \times \mathcal{E} \to \mathcal{T}$. To each source state $s \in S$, we associate a mapping f_s from E to $\mathcal{T}$ defined by $f_s(e) = f(s, e)$, $\forall e \in \mathcal{E}$. Then the family $\{f_s \mid s \in S\}$ completely specifies the underlying A-code $(S, \mathcal{T}, \mathcal{E})$. For our purpose, we shall require that the functions f_s have some additional properties, defined as follows:

Definition 3.1 An A-code $(S, \mathcal{T}, \mathcal{E})$ with the authentication mapping f: $S \times \mathcal{E} \to \mathcal{T}$ is called linear (over F_p) if

1. Both $\mathcal{E}$ and $\mathcal{T}$ are linear spaces over F_p;
2. For each $s \in S$, f_s is F_p-linear from $\mathcal{E}$ to $\mathcal{T}$.

In the sequel, we assume that $(S, \mathcal{T}, \mathcal{E})$ with authentication mapping f is a linear A-code with $\mathcal{E} = V(u, p)$, $\mathcal{T} = V(v, p)$, where p is a prime. Note that for fixed basis of $\mathcal{E}$

and $\mathcal{T}$ each linear mapping from $\mathcal{E}$ to $\mathcal{T}$ can be identified as a $u \times v$ matrix over F_p. Thus we will assume that S is a subset of u by v matrices over F_p, and f_s can be defined as $f_s(e) = es = t$, where $e \in V(u, p)$, $t \in V(v, p)$.

To construct our FSS scheme we use a linear A-code and a one-way function based on the discrete logarithm problem, given by,

$$f_{p,g}: x \to g^x \mod q$$

where p, q are prime and $p|q-1$. The construction works as follows.

- *Prekey Generation*: The centre selects primes q and p such that $p|q-1$ and a cyclic subgroup, H_p, of F_q^* with order p such that the discrete logarithm over H_p is hard. He also chooses two elements $g, h \in H_p$, and publishes (p, q, g, h).
- *Key Generation*: The signer chooses a secret key sk which consists of two authentication keys of a linear A-code $(S, \mathcal{T}, \mathcal{E})$ over F_p.
  That is, $sk = (e, e')$, $e, e' \in \mathcal{E}$, where $e = [e_1, \ldots, e_u]$, $e' = [e'_1, \ldots, e'_u]$, $\forall e_i, e'_j \in F_p$. The corresponding public key pk is $g^e \odot h^{e'}$, is defined as

$$g^e \odot h^{e'} = [\, g^{e_1} h^{e'_1}, \ \ldots, \ g^{e_u} h^{e'_u} \,] \ (\mathrm{mod}\ q)$$
$$= [pk_1, \ldots, pk_u].$$

- *Sign*: To sign a message $s \in S$, the signer applies the authentication code $(S, \mathcal{T}, \mathcal{E})$ to generate two authentication tags t and t' corresponding to the key e and e'. That is, the signature for a message s is

$$(s, f(s, e), f(s, e')) = (s, es, e's) \ (\mathrm{mod}\ p)$$
$$= (s, t, t')$$
$$= (s, [t_1, \ldots, t_v], [t'_1, \ldots, t'_v])$$

- *Test*: For a message

$$s = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,v} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,v} \\ \cdots & \cdots & \cdots & \cdots \\ s_{u,1} & s_{u,2} & \cdots & s_{u,v} \end{pmatrix},$$

  (s, t, t') is a valid signed message iff for all $1 \le i \le v$,

$$g^{t_i} h^{t'_i} = (pk_1)^{s_{1,i}} (pk_2)^{s_{2,i}} \ \ldots \ (pk_u)^{s_{u,i}} \ (\mathrm{mod}\ q).$$

- *Proof of Forgery*: If there is a forged signature $(\tilde{t}, \tilde{t}')$ on a message s, then the presumed signer can produce his own signature on the same message, namely (t, t'), and show that these two signatures collide.

Theorem 3.1 Let ℓ be the security parameter of the underlying discrete logarithm problem. If the linear A-code $(S, \mathcal{T}, \mathcal{E})$ is ϵ-secure, then the above construction results in a (ℓ, δ)-secure FSS scheme, where $\delta = \log(1/\epsilon)$.

Proof: Clearly, if both the signer and the recipient follow the protocol, then every signature generated by the signer will pass the test by the recipient. We are left to show that the scheme is (ℓ, δ)-secure. First, assume that an unbounded forger tries to generate a signature that passes the test with respect to the public key such that the signer cannot provide a proof of forgery. We may further assume that the forger can solve the underlying discrete logarithm. It follows that the forger can calculate $a = \log_g h \in F_p$ and $e + ae' = [e_1+ae'_1, \ldots, e_u + ae'_u]$. Since the forger does not know the authentication keys e and e', the success probability of such a forgery is

$$\begin{aligned} P_{forgery} &= \max_{s\in S}\max_{t,t'\in\tau}\frac{|\{(e,e')\in\varepsilon^2 \mid es=t, e's=t'\}|}{|\{(e,e')\in\varepsilon \mid \text{give} \quad e+ae' \quad \text{and}\, a\}|} \\ &= \frac{|\{e\in\varepsilon \mid es=t\}|}{|\{e\in\varepsilon\}|} \\ &= P_I \\ &\le \epsilon \end{aligned}$$

In fact, we have calculated the value of $|V|$ and $|W|$ as in the general construction of [20].

Next, we show that knowing two different signatures on a message that pass the test, anyone can solve the underlying logarithm problem, i.e., find $a = \log_g h$ or $a^{-1} = \log_h g$. Let $(t, t') = ([t_1, \ldots, t_v], [t'_1, \ldots, t'_v])$ and $(\tilde{t}, \tilde{t}') = ([\tilde{t}_1, \ldots, \tilde{t}_v], [\tilde{t}'_1, \ldots, \tilde{t}'_v])$ be two valid signatures (w.r.t the public key) for a message s. Since $(t, t') \neq (\tilde{t}, \tilde{t}')$, we may assume that there exists i such that $t'_i \neq \tilde{t}'_i$ (or alternatively, we may assume that there is j such that $t_j \neq \tilde{t}_j$), so $t'_i - \tilde{t}'_i \neq 0 \bmod p$. On the other hand, since (t, t') and $(\tilde{t}, \tilde{t}')$ pass the verification test with respect to the common public key $[pk_1, \ldots, pk_u]$, we have

$$g^{t_i} h^{t'_i} = g^{\tilde{t}_i} h^{\tilde{t}'_i} = (pk_1)^{s_{1,i}} \quad \ldots \quad (pk_u)^{s_{u,i}} \mod p .$$

It follows that $h^{(t'_i - \tilde{t}'_i)} = g^{(\tilde{t}_i - t_i)} \bmod q$, and so $g^{\log_g h(t'_i - \tilde{t}'_i)} = g^{(\tilde{t}_i - t_i)} \bmod p$. Therefore $a = \log_g h = (t'_i - \tilde{t}'_i)(\tilde{t}_i - t_i)^{-1} \bmod p$. From this, we conclude that (1) the scheme is secure against the polynomially bounded signer, that is, a signer cannot construct a signature and later provide a proof of forgery, and (2) if there is a forged signature $(\tilde{t}, \tilde{t}')$ on a message s, then the presumed signer can produce his own signature on the same message, namely (t, t'), and show that these two signatures collide, and/or reveal the value $\log_g h$. ❑

Example 3.1 Let $S = \tau = F_p$ and $\varepsilon = F_p \times F_p$.
We define a function f: $S \times \varepsilon \to \tau$, by $f(s, (e_1, e_2)) = e_1 + se_2$. Then it is easy to verify that (S, τ, ε) is a linear A-code over F_p. In terms of matrix representation, we can write a source state s as a 2×1 matrix over F_p

$$S = \{s = \begin{pmatrix} 1 \\ w \end{pmatrix} \mid w \in F_p\},$$

and the authentication mapping f as $f(s, (e_1, e_2)) = (e_1, e_2)\begin{pmatrix} 1 \\ w \end{pmatrix} = e_1 + we_2$. The FSS based on this linear A-code is the same as the *vHP* scheme.

4. CONSTRUCTION OF LINEAR A-CODES

Although the linear A-code in example 3.1 is nearly optimal (i.e., nearly meets the square root bound [9]), it requires that the size of the key be double the size of the source, i.e., $\log |\mathcal{E}| = 2\log |S|$, and so the size of the key grows linearly with the size of the source. In the following, we construct non-optimal linear A-codes in which the size of the source space is much larger than the size of the key space.

A polynomial of the form

$$L(x) = \sum_{i=0}^{n} a_i x^{p^i}$$

with coefficients in an extension field F_{p^m} of F_p is called a p-polynomial over F_{p^m}. If the value of p is fixed once, or is clear from the context, it is also called a *linearised polynomial*. It is well-known that if F is an arbitrary extension field of F_{p^m} and $L(x)$ is a linearised polynomial over F_{p^m}, then

$L(\beta + \gamma) = L(\beta) + L(\gamma)$ for all $\beta, \gamma \in F$,
$L(c\beta) = cL(\beta)$ for all $c \in F_p$ and all $\beta \in F$.

Thus, if F is considered a vector space over F_p, then the linearised polynomial $L(x)$ induces a linear operator on F.

Next we construct a linear A-code from linearised polynomials. We note that linearised polynomials have also been used to construct authentication codes for non-trusting parties [12, 13] and message authentication codes for multiple authentication [19]. Let p be a prime and assume

- $S = \{L_s(x) = \sum_{i=0}^{k-1} a_i x^{p^i} \mid a_i \in F_{p^r}\}$ that is, we let each source state $s \in S$ correspond to a linearised polynomial over F_{p^r}, denoted by $L_s(x)$. We use the linearised polynomials up to degree p^{k-1}, resulting in $(p^r)^k$ different polynomials, and so $|S| = p^{rk}$.
- $\mathcal{E} = \{(e_1, e_2) \mid e_1, e_2 \in F_{p^r}\}$, and so $|\mathcal{E}| = p^{2r}$.
- $T = F_{p^r}$.
- The authentication mapping f: $S \times \mathcal{E} \rightarrow \mathcal{T}$ is defined by $f(L_s(x), (e_1, e_2)) = e_1 + L_s(e_2)$.

Theorem 4.1 The above construction results in a linear A-code $(S, \mathcal{E}, \mathcal{T})$ over F_p with the following parameters

$$|S| = p^{rk}, |\mathcal{E}| = p^{2r}, |\mathcal{T}| = p^r$$

and

$$P_I = p^{-r}, P_S = p^{-(r-k+1)}.$$

Proof: The parameter values for $|S|$, $|\mathcal{E}|$ and $|\mathcal{T}|$ are obvious. We are left to show that the constructed A-code is linear and $P_I = p^{-r}$, $P_S = p^{-(r-k+1)}$. Consider F_{p^r} as a vector space

over F_p, then $\mathcal{E}$ and $\mathcal{T}$ are $2r$-dimensional and r-dimensional vector spaces over F_p, respectively. For each lineralized polynomial $L(x) \in S$ considered as a source state, we show that $f_{L(x)}$ is a F_p-linear mapping from $\mathcal{E}$ to $\mathcal{T}$. Note that the finite field F_{p^r} has prime characteristic p, and we know

$$(a+b)^{p^i} = a^{p^i} + b^{p^i}$$

for $a, b \in F_{p^r}$ and any integer i. It is then straightforward to verify that $f_{L(x)}$ is indeed linear.

For the probability of success of impersonation attack P_I, we have

$$\begin{aligned} P_I &= \max_{L(x)\in S} \max_{t\in\tau} \frac{|\{e\in \varepsilon \mid f_{L(x)}(e)=t\}|}{|\varepsilon|} \\ &= \max_{L(x)\in S} \max_{t\in F_{p^r}} \frac{|\{e\in \varepsilon \mid f_{L(x)}(e)=0\}|}{|\varepsilon|} \text{ (since } f_{L(x)} \text{ is linear)} \\ &= \frac{1}{|\tau|} \text{ (since } f_{L(x)} \text{ is subjective)} \\ &= \frac{1}{p^r} \end{aligned}$$

For the probability of success of substitution attack P_S, we have

$$\begin{aligned} P_S &= \max_{\substack{L(x),L'(x)\in S \\ L(x)\neq L'(x)}} \max_{t,t'\in\tau} \frac{|\{e\in \varepsilon \mid f_{L(x)}(e)=t,\, f_{L'(x)}(e)=t'\}|}{|\{e\in \varepsilon \mid f_{L(x)}(e)=t\}} \\ &= \max_{\substack{L(x),L'(x)\in S \\ L(x)\neq L'(x)}} \max_{t,t'\in\tau} \frac{|\{(e_1,e_2)\in F_{p^r}^2 \mid e_1+L(e_2)=t, e_1+L'(e_2)=t'\}|}{|\{(e_1,e_2)\in F_{p^r}^2 \mid e_1+f(e_2)=t\}|} \\ &= \max_{\substack{L(x),L'(x)\in S \\ L(x)\neq L'(x)}} \max_{t,t'\in\tau} \frac{|\{e_2\in F_{P^r} \mid L(e_2)-L'(e_2)=t-t'\}|}{p^r} \end{aligned}$$

Since $L(x)$ and $L'(x)$ are polynomials of degree at most p^{k-1}, so is $L(x) - L'(x)$, it follows that $L(x) - L'(x)$ has at most p^{k-1} roots and so $|\{e_2 \in F_{P^r} \mid L(e_2) - L'(e_2) = t - t'\}| \le p^{k-1}$. Hence

$$P_S = \frac{p^{k-1}}{p^r} = p^{-(r-k+1)}.$$

So far, we complete the proof of the desired result. ❑

Combining Theorem 3.1 and 4.1, we obtain the following corollary.

Corollary 4.1 Let p and q be primes such that ℓ is the required security parameter so that a polynomially bounded forger cannot break the underlying discrete logarithm problem. Then the above linear A-code results in a (ℓ, δ)-secure FSS scheme such that $\delta = (r - k + 1) \log |p|$.

5. EFFICIENCY MEASURES OF FSS SCHEMES

In this section we compare the efficiency of our proposed scheme with the most efficient FSS scheme, namely vHP. We first fix the level of security provided by the two schemes, and then find the sizes of the secret key, the public key and the signature. Table 1 gives the results of the comparison of FSS schemes when security levels of the receiver and the sender are given by ℓ and δ, respectively. In this comparison, the first two schemes [25, 26] (first and second column of the table) are chosen because they have provable security. The first scheme (referred as vHP scheme in this paper) is the most efficient provably secure scheme, based on the discrete logarithm problems. The third column is an FSS scheme based on RSA [23]. The fourth column is a factorisation based scheme proposed in [24]. Column five corresponds to the scheme from Theorem 4.1 with $r = k$.

In vHP scheme, given the security parameter (ℓ, δ), first $K = \max(\ell, \delta)$ is found and then the prime p is chosen such that $\log p \geq K$. The value of q is chosen such that $p|q - 1$ and $(q - 1)/p$ be upper-bounded by a polynomial in K (page 237 and 238 [18]). Since the sizes of p and q can be independently chosen, we use $\hat{K}$ to denote $\log_2 q$.

In the factorisation scheme of [26], the security level of the sender, δ, satisfies $\tau = \hat{\rho} + \delta$ where τ is the *bundling degree* (which determines the number of secret key pre-images for a particular public key image) and $2^{\hat{\rho}}$ is the size of the message space. Security parameter of the receiver, ℓ, is determined by the difficulty of factoring the modulus n (where $n = pq$ and p and q are both prime numbers). Now for a given pair of security parameters, (ℓ, δ), the size of the modulus N_ℓ is determined by ℓ but determining τ requires knowledge of the size of the message space. Assume $\hat{\rho} = \log_2 p \approx \log_2 q = N_\ell/2$. This means that $\tau = \delta + N_\ell/2$. Now the efficiency parameters of the system can be given as shown in the table. In particular, the sizes of secret and public keys are $2(\tau + N_\ell)$ and $2N_\ell$ respectively.

In RSA-based FSS scheme [23], τ is the bundling degree and is defined as $\tau = \log_2 \varnothing(n)$, and security of the receiver is determined by the difficulty of factoring n (where $n = pq$ and p and q are both prime numbers). This means that $\tau \approx \log_2 n$. To design a system with security parameters (ℓ, δ), first N_ℓ, the modulus size that provides security level ℓ for the receiver is determined and then $K = \max(\delta, N_\ell)$. The modulus n is chosen such that $\log_2 n = K$. With this choice, the system provides adequate security for sender and receiver.

In the factorisation scheme of [24], the security level of the sender is $\log_2 q$. Security level of the receiver is determined by the difficulty of discrete logarithm in Z_P^* and factorisation of n. First N_ℓ, which is the modulus size for which factorisation has difficulty is chosen. Then $K = \max\left(\frac{N_\ell}{2}, \sigma\right)$ is calculated. Since the size of P can be chosen much greater than $\log_2 n$, we use $\hat{K}$ to denote $\log_2 P$.

Table 1. Efficiency parameters comparison.

	vHP[25]	*Fact.*[26]	*RSA*[23]	*Fact.*[24]	*Our scheme*
Message Size	K	K	K	$2K$	r^2K
Length of Secret Key	$4K$	$4K + 2\delta$	$4K$	$4K$	$4rK$
Length of Public Key	$2\hat{K}$	$2K$	$2K$	$2\hat{K}$	$2r\hat{K}$
Length of Signature	$2K$	$2K+\delta$	$4K$	$2K$	$2rK$
Underlying Security Assumption	DL	Fact	Fact	Fact & DL	DL

As pointed out in Example 3.1, if $r = k = 1$, then our scheme coincides with vHP scheme.

Efficiency with respect to the message-length We also need to consider the relative lengths of the message and the signature. If the lengths of the signature and the message are denoted by $|y|$ and $|x|$ respectively, then $\rho = \frac{|x|}{|y|}$ is a measure of communication efficiency of the scheme.

As pointed out in Table 1, in vHP scheme messages are of length $\log_2 p$ and signatures are of length $2\log_2 q$. This means that $\rho = \frac{1}{2}$ and so for every bit of authenticated message, 2 bits of signature are required. In our scheme, messages and signatures are of size $r^2\log p$ and $2r\log p$, respectively and so $\rho = \frac{r}{2}$. Our scheme is the only FSS that allows ρ to change with parameters of the system. Table 2 summaries these results.

Table 2. Comparison of communication efficiency with respect to the message-length.

	vHP[25]	*Fact.*[26]	*RSA*[23]	*Fact.*[24]	*Our scheme*
ρ	$\frac{1}{2}$	$<\frac{1}{2}$	$\frac{1}{4}$	1	$\frac{r}{2}$

Signing long messages To sign long messages one can use (i) our proposed FSS, or (ii) hash-then-sign approach using one of the existing FSS. In the following we compare two schemes: one based on the construction presented in this paper, and the other one using vHP scheme and a provably secure hash function proposed in [3, 5]. The hash function requires on average one multiplication for each bit of the message, and the size of the hash value is equal to the size of the modulus. We assume that the length of the message is $r \times kK$ bits for some integers r and k such that $r \geq k$, $K = \log_2 p$ and $\hat{K} = \log_2 q$.

The table shows that signing using the new construction is $K/2$ times faster, while verification is approximately $rk/2$ times slower. For example, to achieve adequate security [15], we choose $K = 151$ bits and $\hat{K} = 1881$ bits [15]. Also to simplify the comparison, we assume $r = k$. To sign a 1 megabyte message using hash-then sign approach (i.e. using vHP scheme with a secure hash function proposed in [5]), the number of multiplications required for signing and testing are 1,065,458 and 302, respectively. However, by using the proposed approach, the number of multiplications required for signing and testing are 14,112 and 1,090,824, respectively. This asymmetry between the amount of computation required for signing and verification is useful in applications where the signer has limited computing power, for example using a smart card, and the verifier has a powerful server, for example a bank.

Table 3. Complexity of the two FSS approaches for signing long messages.

	Hash-then-sign approach*	Our Scheme
Sign (number of multiplications)	$\approx rkK + 2$	$2rk$
Test (number of multiplications)	$\leq 2K$	$\leq (2 + k)rK$
Length of Secret Key	$4K$	$4kK$
Length of Public Key	$2\hat{K}$	$2r\hat{K}$
Length of Signature	$2K$	$2rK$
Underlying Security Assumption	Collision-Resistant Hash Function & DL	DL

Note: * Hashing uses a provably secure hash function proposed in [3, 5] and signing is by using vHP scheme [25].

6. THRESHOLD FAIL-STOP SIGNATURE SCHEMES

An attraction of using linear A-codes to construct FSS schemes is their flexibility. Because of the algebraic properties of the linear A-codes such FSS, schemes can be easily extended to distributed environments. In the following we show how to construct a threshold FSS scheme based on the proposed linear scheme.

Threshold cryptography, and in particular, threshold signature, was independently invented by Desmedt [6], Boyd [2], Croft and Harris [4]. The main goal of threshold cryptography is to replace a system entity, such as a transmitter, in a classical cryptosystem with a group of entities sharing the same power. A threshold cryptosystem must remain secure not only under the attacks on the original cryptosystem, but also from new types of attacks that are introduced because of the distributed structure of the system.

In a (k, n) threshold signature scheme [8], signature generation requires the collaboration of at least k members of a set of n signers. Although construction of threshold signature schemes generally uses a combination of secret sharing schemes and signature schemes, as noted in [7], a simplistic combination of the two primitives could result in a completely insecure systems that allows the members of an authorized group to recover the secret key of the signature scheme. In a secure threshold signature scheme the power of signature generation must be shared among n signers in such a way that k signers can collaborate to produce a valid signature for any given message, whilst no subset of fewer than k participants can forge a signature.

Similarly, an FSS scheme is called a (k, n) threshold FSS scheme if the role of the signer in an FSS scheme is replaced by n polynomially bounded signers in such a way that key generation, signature generation and proof of forgery require the collaboration of at least k signers. We will present an efficient (k, k) threshold FSS scheme by modifying our previous construction of a single signer. We then discuss the possibility of its extension to the (k, n) case.

6.1 (k, k) Threshold FSS Schemes

We construct a (k, k) scheme with k signers $P_1, \ldots, P_k$. Again we assume that $(S, \mathcal{T}, \mathcal{E})$ with authentication mapping f is a linear A-code. The scheme works as follows:

- *Prekey Generation*: The center selects primes q and p such that $p|q-1$ and a cyclic subgroup H_p of F_q^* with order p such that the discrete logarithm over H_p is hard. He also chooses two elements $g, h \in H_p$, and publishes (p, q, g, h).
- *Key Co-Generation*: Each signer P_i chooses a secret key $sk(i)$ which consists of two authentication keys of a linear A-code $(S, \mathcal{T}, \mathcal{E})$ over F_p, that is, $sk(i) = (e(i), e'(i)), e(i), e'(i) \in \mathcal{E}$. We write

  $e(i) = [e_1(i), \ldots, e_u(i)], e'(i) = [e'_1(i), \ldots, e'_u(i)], \forall e_j(i), e'_j(i) \in F_p$.

  Then P_i broadcasts to other signers

  $$g^{e(i)} \odot h^{e'(i)} = [g^{e_1(i)} h^{e'_1(i)}, \ \ldots, \ g^{e_u(i)} h^{e'_u(i)}] \pmod q$$
  $$= [pk_1(i), \ldots, pk_u(i)].$$

  The group public key is

  $$pk = [\prod_{i=1}^{k} pk_1(i), \ldots, \prod_{i=1}^{k} pk_u(i)].$$

- *Co-Sign*: To sign a message $s \in S$, each signer P_i applies the authentication code $(S, \mathcal{T}, \mathcal{E})$ to generate two authentication tags $t(i)$ and $t'(i)$ corresponding to the key $e(i)$ and $e'(i)$. That is, the (partial) signature for a message s of P_i is

  $$(s, f(s, e(i)), f(s, e'(i))) = (s, e(i)s, e'(i)s) \pmod p$$
  $$= (s, t(i), t'(i))$$
  $$= (s, [t_1(i), \ldots, t_v(i)], [t'_1(i), \ldots, t'_v(i)]),$$

  and is $(s, t(i), t'(i))$ broadcast to other users. The final signature for the group is

  $$(t, t') = ([\prod_{i=1}^{k} t_1(i), \ldots, \prod_{i=1}^{k} t_v(i)], [\prod_{i=1}^{k} t'_1(i), \ldots, \prod_{i=1}^{k} t'_v(i)]) \pmod p.$$

- *Test*: exactly the same as the construction in Section 3 for the single case.
- *Proof of Forgery*: exactly the same as the construction in Section 3 for the single case.

Theorem 6.1 Under the assumption that the discrete logarithm is intractable, the above scheme is (k, k) threshold FSS scheme.

Proof: Completeness and soundness directly follow from the description of the scheme.

We are left to prove that any up to $k - 1$ signers cannot generate a valid signature. The proof of security is by using a simulation argument for the view of the adversary, and showing that an adversary who has access to all the key information of the corrupted signers and the signature on s could generate by itself all the other public information produced by the protocol. Without loss of generality, assume that an adversary $\mathcal{A}$ has

corrupted the first $k-1$ signers $P_1, \ldots, P_{k-1}$ and has learned their secrets. We give a simulator *SIMU* for our scheme. The input to the simulator *SIMU* is the message s and its signature $(s, (t, t'))$. However, the secret information held by P_k is never exposed and is not simulated.

The *SIMU* works as follows.

1. *Key Co-Generation*
   - Choose $(\tilde{e}(1), \tilde{e}'(1)), \ldots (\tilde{e}(k-1), \tilde{e}'(k-1)) \in_R \varepsilon \times \varepsilon$, where

$$\tilde{e}(i) = [\tilde{e}_1(i), \ldots, \tilde{e}_u(i)], \tilde{e}'(i) = [\tilde{e}_1'(i), \ldots, \tilde{e}_u'(i)], \forall \tilde{e}_j(i), \tilde{e}_j'(i) \in F_p, \forall 1 \le i \le k-1;$$

   - Compute the 'broadcast' key of P_i, $\forall 1 \le i \le k-1$:

$$\begin{aligned} g^{\tilde{e}(i)} \odot h^{\tilde{e}'(i)} &= [g^{\tilde{e}_1(i)} h^{\tilde{e}_1'(i)}, \ \ldots, \ g^{\tilde{e}_u(i)} h^{\tilde{e}_u'(i)}] \\ &= [\tilde{pk}_1(i), \ldots, \tilde{pk}_u(i)]. \end{aligned}$$

   - Set

$$g^{\tilde{e}(k)} \odot h^{\tilde{e}'(k)} = [\prod_{i=1}^{k} pk_1(i) / \prod_{i=1}^{k-1} \tilde{pk}_1(i), \ldots, \prod_{i=1}^{k} pk_u(i) / \prod_{i=1}^{k-1} \tilde{pk}_u(i)].$$

At the end the simulation in this phase, each P_i, $1 \le i \le k-1$, holds a 'simulation' version of the execution of the protocol of Key Co-Generation. We use notation $\tilde{a}$ in the simulation corresponding to a in the execution of the protocol.

2. *Co-Sign*:
   - Compute the partial signatures

$$(s, \tilde{t}(i), \tilde{t}'(i)) = (s, f(s, \tilde{e}(i)), f(s, \tilde{e}'(i))) = (s, [\tilde{t}_1(i), \ldots, \tilde{t}_v(i)], [\tilde{t}_1'(i), \ldots, \tilde{t}_v'(i)]),$$

     for all $1 \le i \le k-1$.

   - Set

$$\begin{aligned} \tilde{t}(k) &= ([\prod_{i=1}^{k} t_1(i) / \prod_{i=1}^{k-1} \tilde{t}_1(i), \ldots, \prod_{i=1}^{k} t_v(i) / \prod_{i=1}^{k-1} \tilde{t}_v(i)], \\ \tilde{t}'(k) &= [\prod_{i=1}^{k} t_i'(i) / \prod_{i=1}^{k-1} \tilde{t}_1'(i), \ldots, \prod_{i=1}^{k} t_v'(i) / \prod_{i=1}^{k-1} \tilde{t}_v'(i)]). \end{aligned}$$

It is straightforward to verify that the view of adversary $\mathcal{A}$ on execution of the protocol, and its view on execution of *SIMU* are statistically indistinguishable, and the result follows. ❑

In the above (k, k) scheme, the size of each signer's key, the size of the public key and the length of the signature are exactly the same as the single signer scheme and so is very efficient.

6.2 Extensions

Proof of forgery In the above scheme, proving forgery requires collaboration of all the signers. That is, each signer must submit his partial signature and the correct signature of the signers on the message must be constructed. By showing two different signatures that pass the verification test, the signers can prove that a forgery has happened. All the signers must honestly participate in proving forgery, and if only one of them does not submit his correct partial signature the forgery cannot be proved. An open question is how to design systems in which the forgery can be proved if some of the users do not take part in the proof of forgery. The best example is when a single honest signer can provide proof of forgery.

(*k*, *n*) Schemes A natural extension of (*k*, *k*) schemes are (*k*, *n*) schemes where collaboration of *k* out of *n* signers is required. Specifically, any *k* out of *n* signers can generate a signature and prove forgery if a forged signature is found. Using the standard approach in threshold cryptography, we can convert an FSS scheme to a (*k*, *n*) FSS by splitting the secret key of the FSS into *n* signers using a (*k*, *n*) secret sharing scheme. The system can be designed with or without a trusted third party. If we assume a secure channels exists between participants, then the (*n*, *n*) scheme in this paper can be converted to a (*k*, *n*) scheme using cumulative arrays [11].

7. CONCLUSIONS

We introduced a new class of A-codes called linear A-codes, and used it to construct an FSS which allows efficient signature generation of long messages. An important property of the FSS scheme is that ρ is a design parameter of the system and so by designing systems with small ρ, arbitrary length messages can be directly signed without the need to use a hash function.

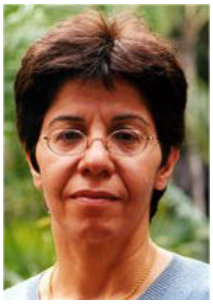
We also showed that it is possible to use the code to construct a distributed FSS system. We described a construction for a (*k*, *k*) thresholds scheme and discussed its extension to (*k*, *n*), $n > k$ scheme.

REFERENCES

1. N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," *Advances in Cryptology – Eurocrypt '97*, Lecture Notes in Computer Science 1233, 1997, pp. 480-494.
2. C. Boyd, "Digital multisignatures," *Cryptography and Coding*, Beker and Piper eds., Clarendon Press, 1989, pp. 241-246.
3. D. Chaum, E. van Heijst, and B. Pfitzmann, "Cryptographically strong undeniable signatures, unconditionally secure for the signer," *Interner Bericht, Fakultät für Informatik*, 1/91, 1990.
4. R. A. Croft and S. P. Harris, "Public-key cryptography and re-usable shared secrets," *Cryptography and Coding*, Beker and Pipereds ed., Clarendon Press, 1989, pp. 241-246.

5. I. B. Damgård, "Collision free hash functions and public key signature scheme," Lecture Notes in Computer Science 304, 1988, pp. 203-216.
6. Y. Desmedt, "Society and group oriented cryptology: a new concept," *Advances in Cryptography–CRYPTO* '87, Lecture Notes in Computer Science 293, 1988, pp. 120-127.
7. Y. Desmedt, "Threshold cryptography," *European Transactions on Telecommunications*, Vol. 5, No. 4, 1994, pp. 449-457.
8. Y. Desmedt and Y. Frankel, "Homomorphic zero-knowledge threshold schemes over any finite group," *SIAM Journal Discrete Mathematics*, Vol. 7, No. 4, 1994, pp. 667-679.
9. E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane, "Codes which detect deception," *The Bell System Technical Journal*, Vol. 33, No. 3, 1974, pp. 405-424.
10. O. Goldreich, *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, Springer, 1999.
11. W. A. Jackson and K. M. Martin, "Cumulative arrays and geometric secret Shaing schemes," *Advances in Cryptology–Auscrypt* '92, Lecture Notes in Computer Science 718, 1993, pp. 48-55.
12. T. Johansson, "Contributions to unconditionally secure authentication," Ph.D. thesis, Lund, Dept. of Information Technology, Lund University, 1994.
13. T. Johansson, "Authentication codes for nontrusting parties obtained from rank metric codes," *Designs, Codes and Cryptography*, Vol. 6, 1995, pp. 205-218.
14. L. Lamport, "Constructing digital signatures from a one-way function," *PSRI International CSL*-98, 1979.
15. A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," online: http://www.cryptosavvy.com, Extended abstract appeared in Commercial Applications, *Price Waterhouse Coopers, CCE Quarterly Journals*, Vol. 3, 1999, pp. 3-9.
16. T. P. Pedersen and B. Pfitzmann, "Fail-stop signatures," *SIAM Journal on Computing*, Vol. 26, No. 2, 1997, pp. 291-330.
17. B. Pfitzmann, "Fail-stop signatures: Principles and applications," in *Proceedings of Compsec* '91, *8th World Conference on Computer Security, Audit and Control*, 1991, pp. 125-134.
18. B. Pfitzmann, "Digital signature schemes-General framework and fail-stop signatures," Lecture Notes in Computer Science 1100, Springer-Verlag, 1996.
19. R. Safavi-Naini, S. Bakhtiari, and C. Charnes, "MRD hashing," in *Proceedings of Fast Software Encryption Workshop*, Lecture Notes in Computer Science 1372, 1998, pp. 134-149.
20. R. Safavi-Naini and W. Susilo, "A general construction for fail-stop signature using authentication codes," in *Proceedings of Workshop on Cryptography and Combinatorial Number Theory* (*CCNT* '99), 2001, pp. 343-356.
21. R. Safavi-Naini and W. Susilo, "Fail-stop threshold signature schemes based on discrete logarithm and factorization," *The Third International Workshop on Information Security–ISW* 2000, Lecture Notes in Computer Science 1975, 2000, pp. 292-307.
22. G. J. Simmons, "Authentication theory/coding theory," *Advances in Cryptology–Crypto* '84, Lecture Notes in Computer Science 196, 1984, pp. 411-431.
23. W. Susilo, R. Safavi-Naini, and J. Pieprzyk, "RSA-based fail-stop signature schemes," *International Workshop on Security* (*IWSec* '99), IEEE Computer Society

Press, 1999, pp. 161-166.
24. W. Susilo, R. Safavi-Naini, M. Gysin, and J. Seberry, "An efficient fail-stop signature schemes," *The Computer Journal*, Vol. 43, Issue 5, 2000, pp. 430-437.
25. E. van Heijst and T. Pedersen, "How to make efficient fail-stop signatures," *Advances in Cryptology–Eurocrypt* '92, 1992, pp. 337-346.
26. E. van Heijst, T. Pedersen, and B. Pfitzmann, "New constructions of fail-stop signatures and lower bounds," *Advances in Cryptology–Crypto* '92, Lecture Notes in Computer Science 740, 1993, pp. 15-30.
27. M. Waidner and B. Pfitzmann, "The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability," *Advances in Cryptology–Eurocrypt* '89, Lecture Notes in Computer Science 434, 1990, pp. 690.



Rei Safavi-Naini is a Professor of Computer Science at the School of Information Technology and Computer Science of the University of Wollongong. She has received a Ph.D. in Electrical Engineering from University of Waterloo and has worked on design, analysis and implementation of secure systems for the last 16 years.

Willy Susilo has received a Ph.D. in Computer Science from University of Wollongong, Australia. Currently, he is a Lecturer of Computer Science at the School of Information Technology and Computer Science of the University of Wollongong. His research interests include cryptography, computer network security and threshold cryptography.

Huaxiong Wang received his Ph.D. degrees in Mathematics from University of Haifa, Israel and in Computer Science from University of Wollongong, Australia. He is currently with School of Information Technology and Computer Science, University of Wollongong. Prior to his current position, he worked at Fujian Normal University (China), Kobe University (Japan) and National University of Singapore (Singapore). His research interests include cryptography, information security, coding theory and combinatorics.