A General Construction for Fail-Stop Signatures using Authentication Codes

Rei Safavi-Naini and Willy Susilo

Abstract. Security of an ordinary digital signatures rely on a computational assumption. Fail-stop signature schemes provide security for a sender against a forger with unlimited computational power by enabling the sender to provide a proof of forgery, if it occurs. In this paper, we give a general method of constructing fail-stop signature schemes from authentication codes. We also give an example construction, and prove its security.

1. Introduction

Digital signatures, introduced in [4], are the most important cryptographic primitive for providing authentication in electronic world. The original definition of digital signature was subsequently revised [5] to ensure security against a more stringent type of attack known as *adaptive chosen plain-text attack*. Despite stronger requirement, security in digital signature schemes is in a computational sense and hence an enemy with unlimited computing power can always successfully forge a signature. We refer to this type of signatures as *ordinary* signatures.

In an ordinary signature scheme if a forgery occurs, there is no way for the sender to show that a forgery has occurred. This is unavoidable as if the signer is allowed to disavow a forged signature, there is no way of distinguishing between a forged signature from one generated by the signer, and so the signer has always the opportunity of disavowing his own signature. Hence the security for the signer is computational and if the underlying computational assumption is broken a forged signature can be successfully created. On the other hand the security of the receiver is unconditional as verification is a public process.

Fail-stop signature (FSS) schemes were proposed [15, 12, 8] to provide protection against forgeries of an enemy with unlimited computational power. In a FSS there are a number of participants: (i) a polynomially bounded *signer* who signs a message that is verifiable by everyone with access to his public key and is protected against forgery by an unbounded enemy, (ii) one or more polynomially bounded *recipients* who directly, or indirectly through a trusted centre, take part in the key generation process and are protected against repudiation of the signer, and (iii) a *trusted centre* who is trusted by the recipients and only takes part in the key generation phase. There is another group of participants, the so-called *risk-bearers*, such as insurance companies, who will bear a loss if a proof of forgery is accepted and hence a signature is invalidated. For simplicity we do not make any distinction between a recipient and a risk bearer. In the case of forgery, the presumed signer can provide a proof that a forgery has happened. This is by showing that the underlying computational assumption of the system is broken. The system will be stopped at this stage- hence the name *fail-stop*. In this way, a polynomially bounded signer can be protected against a forger with unlimited computational power. It can be shown that (Theorem 3.2 [8]) a secure FSS can be used to construct an ordinary digital signature that is secure in the sense of [5] and so a fail-stop signature scheme provides a stronger notion of security.

A FSS in its basic form is a *one-time digital signature* that can only be used for signing a single message. However, it is possible to extend a FSS scheme to be used for signing multiple messages [2, 14, 10, 1].

1.1. Our Contributions

In this paper, we describe a method of constructing a FSS from an authentication code. We also show an example construction and prove its correctness and security.

The paper is organised as follows. In the next section, we present the notations and review basic concepts of FSS and authentication codes. We briefly examine the previous works on FSS and recall a general construction proposed in [8]. In section 3, we propose a general method of constructing a FSS from an authentication code. In section 4, we give an example of FSS scheme based on our framework and prove its security requirements. Section 5 concludes the paper.

2. Notations and Preliminaries

2.1. Notations

The length of a number n is the length of its binary representation and is denoted by $|n|$. Concatenation of two binary strings x and y is denoted by $x||y$.

The ring of integers modulo a number n is denoted by Z_n, and the multiplicative subgroup of integers relatively prime to n, by Z_n^*. Let N denote the set of natural numbers.
Notation $x \overset{?}{=} y$, means that equality of x and y must be checked.

2.2. Authentication Codes

Authentication codes are used to provide protection against tampering with the messages communicated between two participant over an insecure channel.

An *authentication code (A-code)* is a set of functions $\mathbf{E}$, each from a set of messages $\mathbf{M}$ to a set of codewords $\mathbf{C}$ (also called *authenticated messages*). $\mathbf{E}$ is indexed by a piece of information, $k \in \mathcal{K}$, called *key*. A key k uniquely determines an encoding function e_k and conversely, an encoding function is associated with a unique key. For the sake of brevity and when it is clear from the context the term key may be used instead of its corresponding function.

In a systematic Cartesian authentication code, a codeword c is the concatenation of a message m and a tag t, that is: $c = m||t$. A Cartesian A-code with a message space **M**, a tag space **T** and key space **E** is denoted by $A(M, T, E)$ and can be described by a $|E| \times |M|$ matrix over T. A row of the matrix labelled by a key k, defines the *encoding function*, e_k, from M to T given by $e_k(m) = t$ if $A(k, m) = t$.

We considered two types of attacks where in both the enemy is an *intruder-in-the-middle*. The attacks are:

- **Impersonation Attack**: The enemy introduces a message (m, t), where $m \in \mathcal{M}$ and $t = e_k(m)$, and hopes that this message is accepted as authentic by the receiver.
- **Substitution Attack**: After observing an authentic message (m, t) in the channel, the enemy constructs a message (m', t'), $m \neq m'$, and hopes that it is accepted as authentic by the receiver.

The chance of enemy's success in making the receiver to accept his fraudulent message as authentic when he follows his optimal strategy in impersonation or substitution, is denoted by Pd_0 and Pd_1, respectively.

We assume that the authentication code, the two probability distributions on **M** and **E** are publicly known but the actual value of the key, and so the encoding function is not known.

2.3. Fail-Stop Signature Schemes

Similar to an ordinary digital signature scheme, a fail-stop signature scheme consists of one polynomial time protocol and two polynomial time algorithms.

1. *Key generation*: is a two party protocol between the signer and the centre to generate a pair of *secret key*, s_k, and *public key*, p_k. This is different from ordinary signature schemes where key generation is performed by the signer individually and without the involvement of the receiver.
2. *Sign*: is the algorithm used for signature generation. For a message m and using the secret key s_k, the signature is given by $y = sign(s_k, m)$.
3. *Test*: is the algorithm for testing acceptability of a signature. For a message m and signature y, and given the public key p_k, the algorithm produces an *ok* response if the signature is acceptable under p_k. That is $test(p_k, m, y) \overset{?}{=} ok$.

A FSS also includes two more polynomial time algorithms:

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

A secure fail-stop signature scheme must satisfy the following properties [13, 11, 8].

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 an 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 can generate all the signatures but 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.

FSS are studied in two different models. The main difference between the models is the existence of a dealer who is trusted by all the recipients. Schemes with a trusted dealer, for example [14, 3], allow public verification of the signature and use a two-party protocol between the signer and the dealer to generate the required keys. This ensures that the signer cannot later deny his own signature and can provide a proof of forgery when needed. Schemes without a trusted dealer, for example [13], are obtained by giving the role of the trusted dealer to one of the recipients. This results in a more efficient key exchange at the expense of loosing public verifiability property of the signature. The model is useful for applications such as electronic payment where verification is required for only a single receiver.

Security of a FSS is broken if 1) a signer can construct a signature that he can later prove to be a forgery, 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 completely independent and so two different security parameters, k and σ, are used to show the level of security against the two types of attacks. More specifically, k is the security level of the recipient against the forgery of the signer, and σ is that of the signer against unbounded forger. It is proved [8] that a secure FSS is secure against adaptive chosen plain-text attack and for all $c > 0$ and large enough k, success probability of a polynomially bounded forger is bounded by k^{-c}. For a FSS with security level σ for the signer, the success probability of an unbounded forger is limited by $2^{-\sigma}$.

In the following we briefly recall the general construction given in [8] and outline its security properties.

2.4. The General Construction of FSS

The construction is for a single-message fail-stop signature and uses *bundling homomorphisms*. Bundling homomorphisms can be seen as a special kind of hash functions.

Definition 2.1. [8] *A bundling homomorphism h is a homomorphism $h : G \rightarrow H$ between two Abelian groups $(G, +, 0)$ and $(H, \times, 1)$ that satisfies the following.*

1. *Every image $h(x)$ has at least 2^τ preimages. 2^τ is called* bundling degree *of the homomorphism.*
2. *It is infeasible to find collisions, i.e., two different elements that are mapped to the same value by h.*

To give a more precise definition, we need to consider two families of groups, $\mathbf{G} = (G_K, +, 0)$ and $\mathbf{H} = (H_K, \times, 1)$, and a family of polynomial-time functions indexed by a key, K. The key is determined by the application of a key generation algorithm $g(k, \tau)$, on two input parameters k and τ. The two parameters determine the difficulty of finding collision and the bundling degrees of the homomorphisms, respectively. Given a pair of input parameters, $k, \tau \in N$, firstly, using the key generation algorithm, a key K is calculated and then, G_K, H_K and h_K are determined. For a formal definition of bundling homomorphisms see Definition 4.1 [8].

A bundling homomorphism can be used to construct a FSS scheme as follows. Let the security parameters of the FSS be given as k and σ. The bundling degree of the homomorphism, τ, will be obtained as a function of σ as shown below.

1. *Prekey generation*: The centre computes $K = g(k, \tau)$ and so determines a homomorphism h_K, and two groups G_K and H_K. Let $G = G_K$, $H = K_K$ and $h = h_K$.
2. *Prekey verification*: The signer must be assured that K is a possible output of the algorithm $g(k, \tau)$. This can be through providing a zero-knowledge proof by the centre or by testing the key by the signer. In any case the chance of accepting a *bad* key must be at most $2^{-\sigma}$.
3. *Main key generation gen_A*: the signer generates her secret key $sk := (sk_1, sk_2)$ by choosing sk_1 and sk_2 randomly in G and computes $pk := (pk_1, pk_2)$ where $pk_i := h(sk_i)$ for $i = 1, 2$.
4. The message space M is a subset of Z.
5. *Signing*: The signature on a message $m \in M$ is,
$$s = sign(sk, m) = sk_1 + m \times sk_2$$
where multiplying by m is m times addition in G.
6. *Testing the signature*: can be performed by checking,
$$pk_1 \times pk_2^m \overset{?}{=} h(s)$$
7. *Proof of forgery*: Given an acceptable signature $s' \in G$ on m such that $s' \neq sign(sk, m)$, the signer computes $s := sign(sk, m)$ and $proof := (s, s')$.

8. *Verifying proof of forgery*: Given a pair $(x, x') \in G \times G$, verify that $x \neq x'$ and $h(x) = h(x')$.

Theorem 4.1 [8] proves that for any family of bundling homomorphisms and any choice of parameters the general construction:

1. produces correct signature;
2. a polynomially bounded signer cannot construct a valid signature and a proof of forgery;
3. if an acceptable signature $s^* \neq sign(sk, m^*)$ is found the signer can construct a proof of forgery.

Moreover for the chosen parameters k and σ, a good prekey K and two messages $m, m^* \in M$, with $m \neq m^*$, let

$$T := \{d \in G | h(d) = 1 \wedge \ (m^* - m)d = 0\} \qquad (1)$$

Theorem 4.2 [8] shows that given $s = sign(sk, m)$ and a forged signature $s^* \in G$ such that $test(pk, m^*, s^*) = ok$, the probability that $s^* = sign(sk, m^*)$ is at most $|T|/2^{\tau}$ and so the best chance of success for an unrestricted forger to construct an undetectable forgery is bounded by $|T|/2^{\tau}$. Thus to provide the required level of security σ, we must choose $|T|/2^{\tau} \leq 2^{-\sigma}$.

This general construction is the basis of all known *provably secure constructions* of FSS. It provides a powerful framework by which proving security of a scheme is reduced to specifying the underlying homomorphism, and determining the bundling degree and the set T.

Other Previous Works

The first construction of fail-stop signature [15] uses a one-time signature scheme (similar to [6]) and results in bit by bit signing of the message and so is very impractical. In [9] 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 communication cost. The size of the signature is twice the length of the message.

In [14], an efficient FSS that uses the difficulty of the discrete logarithm problem as the underlying assumption is presented. In the case of a forgery, the presumed signer can solve an instance of the discrete logarithm problem, and prove that the underlying assumption is broken. This is the most efficient scheme known so far which requires only two multiplications for signature generation and results in a signature which is twice the size of the message.

In [8, 11], a formal definition of FSS schemes is given and a general construction using *bundling homomorphism* is proposed. The important property of this construction is its provable security. An instance of this construction uses the difficulty of factoring as the underlying computational assumption of the system [13].

It is also proved that for a system with security level σ for the signer, the signature length and the length of secret key required for signing N messages are at least $2\sigma - 1$ and $(N+1)(\sigma - 1)$, respectively.

3. Fail-stop signatures from A-codes

We are interested in families of A-codes. A family of A-code is defined by a family of message spaces, $\mathcal{M}$, a family of tag spaces, $\mathcal{T}$ and a family of key spaces $\mathcal{E}$. Each of the three families is an infinite collection of sets indexed by $k \in N$. That is, $\mathcal{M} = \{M_k : k \in N\}$, $\mathcal{T} = \{T_k : k \in N\}$ and $\mathcal{E} = \{E_k : k \in N\}$. Moreover, $A(M_k, T_k, E_k)$ is an A-code with $P_d^{(k)} = \max\{pd_0^{(k)}, pd_1^{(k)}\} \leq \epsilon$ where pd_0^k, pd_1^k are probability of success against impersonation and substitution attack in $A(M_k, T_k, E_k)$.

A function h from E to E' is called a *bundling hash function* of degree 2^μ, if it satisfies the following properties:

1. for every $e' \in E'$ which is the image of some $e \in E$, that is $e' = h(e)$, there are at least 2^μ preimages, $e_1, e_2, \cdots e_{2^\mu}$, such that $h(e_i) = e'$, $i = 1 \cdots 2^\mu$.
2. for any $e' \in E'$ with $e' = h(e)$ for some $e \in E$, it is hard to find $e_1 \in E$, $e_1 \neq e$ such that $h(e) = h(e_1)$.
3. for any $e' \in E'$ with $e' = h(e)$ for some $e \in E$, it is difficult to find e'' such that $h(e'') = h(e')$.

To make the second and third requirements precise we need a family of functions indexed by a key K. A key K determines the bundling degree μ and computational difficulty of collision finding, given by τ. This definition has similarities with the definition of bundling homomorphism but is more general and does not require E and E' to have group structure.

Now consider two families of A-codes $\mathcal{A} = \{A(M_k, T_k, E_k) : k \in N\}$ and $\mathcal{A}' = \{A'(M_k, T_k', E_k') : k \in N\}$, a family of polynomial time bundling hash function $\mathcal{H} = \{h_k : k \in N\}$ where $h_k : E_k \mapsto E_k'$, and a family of polynomial time functions $\mathcal{G} = \{g_k : k \in N\}$, where $g_k : T_k \mapsto T_k'$.
We require the following property:

- for any choice of the index k, and for an arbitrary $e \in E_k$ the following is satisfied for all $m \in M_k$:

$$\text{if} \quad e(m) = t, \;\text{ and } h_k(e) = e', \text{then } e'(m) = t' \;\text{ and } g_k(t) = t'$$

Given a six tuple $(\mathcal{A}, \mathcal{A}', \mathcal{E}, \mathcal{E}', \mathcal{H}, \mathcal{G})$ we can construct a FSS as follows.

The index K is the pre-key and is determined by a pre-key generation algorithm $u()$ which takes the following parameters as input: (i) μ, the bundling degree of the hash function, (ii) τ the difficulty of finding collision, and (iii) two security parameters ϵ_k and ϵ_k', for $A(M_K, T_K, E_K)$ and $A'(M_K, T_K', E_K')$, respectively. The resulting index is the pre-key that determines various parameters of the system. The signer must be sure that the K is a possible output of $u()$ by performing a *prekey verification* algorithm.

Once K is determined, $A(M_K, T_K, E_K)$, $A'(M_K, T'_K, E'_K)$, g_K and h_K are fixed and we have the following stages. Let $\epsilon_K = \epsilon$ and $\epsilon'_K = \epsilon'$.

1. *Main key generation*: the signer chooses $e \in E_K$ as his secret key (encoding function) and constructs $e' = h_K(e)$ as his public key (verification function).
2. *Signing*: the signature for the message $m \in M_K$ is given by $t = e(m)$.
3. *Testing of the signature*: a signature t on a message m is verified if $g_K(t) = e'(m)$.
4. *Proof of forgery*: given an acceptable signature t_1 on m where $t_1 \neq e(m)$, the signer produces $t = e(m)$ as the proof of forgery.

Theorem 3.1. *The above FSS satisfies security requirements of an FSS in which the success chance of an unbounded forger is given by*

$$\max\{\max_{e' \in E'} \epsilon(e'), \epsilon, \epsilon'\}$$

while the chance of success for a bounded forger is given by τ.

Proof: As noted earlier we need to prove security of the system against two types of enemy.
Security against an unbounded forger:
An enemy with unlimited computational power wants to forge a signature. He may use his knowledge of a signed message to improve his success chance. There are three possible way of constructing the forged signature: (i) finding the secret key: knowing the function h and the public key e' he can find 2^{μ} pre-images $e \in E_K$ and guessing the correct key has success chance of $2^{-\mu}$. (ii) using impersonation or substitution attack on $A(M, T, E)$ or $A'(M, T', E')$: this gives him the success chance of most ϵ and ϵ'. (iii) combining the two attacks: that is first restricting the key space E to those keys e that satisfy $h(e) = e'$, and then using the restricted A-code obtained from $A(M, T, E)$ for impersonation or substitution attack. Now, let

$$\begin{aligned} v_0 &= \max_{t} \frac{|\{e \in E : e(m) = t\}|}{2^{\mu}} \\ v_1 &= \max_{t,t'} \frac{|\{e \in E : h(e) = e', e(m) = t, e(m') = t'\}|}{|\{e \in E : h(e) = e', e(m) = t\}|} \end{aligned}$$

denote the best success chance in impersonation and substitution attack in the restricted code. Let $\epsilon(e')$ be defined as,

$$\epsilon(e') = \max\{v_0, v_1\}$$

Then the chance of the unbounded forger in this attack is given by

$$P \leq \max_{e' \in E'} \epsilon(e')$$

It is easy to see that $\max_{e' \in E'} \epsilon(e') \geq 2^{-\mu}$. Therefore, the best success probability of an unbounded forger is limited by

$$\max\{\max_{e' \in E'} \epsilon(e'), \epsilon, \epsilon'\}$$

Security against a bounded forger:
The second type of attack is from a bounded signer who attempts to construct a signature that can be refuted later. For this he must find two signatures on the same message m that satisfy the equations: $g(e_1(m)) = e'_1(m)$ where $e'_1 = h(e_1)$. He can success if he can find two secret keys e_1 and e_2 that collide under the hash function h: that is $h(e_1) = h(e_2)$. Such a pair of keys can be used to generate two signatures $e_1(m)$ and $e_2(m)$ which both result in $e'_1 = h(e_1) = h(e_2)$. The difficulty of finding a collision for the hash function will limit his success chance to τ.

□

Using the above construction, constructing an FSS with security parameters (σ, τ) requires choosing σ to satisfy

$$\max\{\max_{e' \in E'} \epsilon(e'), \epsilon, \epsilon'\} \leq 2^{-\sigma}$$

4. A Construction

In the following we describe an example construction for the above model. We will have two families of A-codes with a bundling hash function that satisfy the requirements of the construction above.

Let p and q denote two large primes, $p < q < 2p$, $n = pq$, and $P = 2pq + 1$ where P is also prime. Let g be an element of Z_P^* of order p, $ord_P(g) = p$. Define an A-code as follows.
$\mathbf{M} \in Z_n$.
$\mathbf{T} \in Z_n^*$.
$\mathbf{E} = \{e_{i,j} : 0 \leq i \leq n-1, 0 \leq j \leq n-1\}$ where $e_{ij} : M \mapsto T$ and $e_{ij}(l) = i + jl \pmod{n}$.

Theorem 4.1. *For the above A-code we have* $Pd_0 = \frac{1}{pq}$ *and* $Pd_1 = \frac{1}{p}$.

Proof: (sketch)

1. $Pd_0 = \frac{1}{pq}$.
   We need to show that for an arbitrary source state l, and an arbitrary tag value t, the number of keys e_{ij} such that $e_{ij}(l) = t$ is pq.
   Such keys satisfy the following equation
   $$i + jl = t \pmod{pq}$$
   Since there are pq choices for j and for each choice there is a unique i, then we will obtain pq keys that satisfy the condition.
2. $Pd_1 = \frac{1}{p}$.
   We know that for an arbitrary source state l, a tag t, $0 \leq i < pq$, occurs exactly pq times. Now, consider two source states l and l', and all the keys that produce the tag t for the message l. That is: $i + jl = t$. From these keys, the number of keys (i, j) that produce the tag t' for the message l'

is given by the number of solutions to the equation $i + jl' = t'$, $t' \in Z_{pq}$. Or the number of solutions to,

$$t - t' = j(l - l') \pmod{pq}$$

Consider two cases:

Case 1. $gcd(l - l', pq) = 1$.
Then, $j = (l - l')^{-1}(t - t') \pmod{pq}$, and so there is a unique key with this property.

Case 2. $gcd(l - l', pq) \neq 1$.
So, there can be either
i. $l - l' = kp$, $k \in Z_q \setminus 0$ (non-zero elements of Z_q), or
ii. $l - l' = kq$, $k \in Z_p \setminus 0$ (non-zero elements of Z_p).
We consider each case as follows:
i. Let $l - l' = kp$. Then, $k = 1, 2, \cdots, p, p + 1, \cdots, q - 1$, and so
a. if $k \neq p$, then we have $gcd(k, p) = 1$. This means: $(t - t') = jkp \pmod{pq}$. For a fixed $t - t'$, the number of solutions (j) for this equation is p. This is true because if j_0 satisfies $t - t' = j_0kp \pmod{pq}$, then $(j_0 + uq) \pmod{pq}$ will also satisfy the equation. This is:

$$\begin{aligned} t - t' &= (j_0 + uq)kp \\ &= j_o kp \pmod{pq} \end{aligned}$$

Now, u can take p values, $0, 1, \cdots, p - 1$, and so there are p solutions. This means that a pair (t, t') will occur p times.
b. if $k = p$, we have $(t - t') = jp^2 \pmod{pq}$. For a fixed $(t - t')$, there are also p solutions for this equation.
ii. Let $l - l' = kq$, then $k = 1, 2, \cdots, p - 1$, and hence $gcd(k, q) = 1$. This means: $(t - t') = jkq \pmod{pq}$. For a fixed $(t - t')$, the number of solutions for this equation is q. This is true because if j_0 satisfies $t - t' = j_0kq \pmod{pq}$, then $(j_0 + up) \pmod{pq}$ will also satisfy the equation. Now, u can take q values, $0, 1, \cdots, q - 1$, and so there are q solutions. This means that a pair (t, t') will occur q times.

Hence, the maximum number of solutions for $t - t' = j(l - l') \pmod{pq}$ is q (since $p < q$). Therefore, the probability of success in substitution attack is:

$$\begin{aligned} Pd_1 &= \max_{l,l',t,t'} \frac{|\{e_{ij} : e_{ij}(l) = t, e_{ij}(l') = t'\}|}{|\{e_{ij} : e_{ij}(l) = t\}|} \\ &= \frac{q}{pq} \\ &= \frac{1}{p} \end{aligned}$$

□

Bundling Hash Function

Let H_p be the subgroup of Z_P^* generated by g. We have $|H_p| = p$. We define a mapping $h : Z_n \mapsto H_p$ given by $h(x) = g^x \pmod{P}$. In the following we show that h is a bundling hash function.

1. *For any $x \in H_p$, there are q preimages, y such that $h(y) = x$.*
   This is true because for all elements $y' \in Z_n$ where $y' = y + tp$, and $t \in \{0, 1, ..q-1\}$, we have
$$h(y') = g^{y+tp} = g^y = x$$
2. *Given $x \in H_p$ it is difficult to find a y such that $h(y) = x$.*
   This is true because finding y that satisfies $g^y = x \pmod{P}$ is equivalent to finding discrete logarithm in group H_p which is know to be hard (In fact, solving discrete logarithm in group H_p is considerably more difficult than factoring n [7]).
3. *Knowing x and y that satisfy $g^y = x \pmod{P}$, finding y' such that $h(y) = h(y')$ is equivalent to factoring n.*
   This is true because $h(y) = h(y')$ implies $g^y = g^{y'} \pmod{P}$ and because g is of order p, $y = y' \pmod{p}$ and so $y - y'$ is a multiple of p.

The above bundling hash function partitions Z_n into p subsets each of size q. That is, $Z_n = V_0 \cup V_1 \cdots \cup V_{p-1}$ where $V_i = i + tp$, $0 \leq t \leq q-1$. From above, we know that an element x of H_p corresponds to a unique V_i. We use $H_p^{(i)}$ to denote x.

Now consider a family of A-codes, $A'(M, T', E')$ where $M = Z_n$, $T' = H_p$, $E' = H_p \times H_p$ and $e'_{IJ}(l) = g^i g^{jl}$ where $I = H_p^{(i)}$ and $J = H_p^{(j)}$ respectively.

Deception Probabilities for $A'(M, T', E')$

Theorem 4.2. *In $A'(M, T', E')$ we have $Pd_0 = \frac{1}{p}$ and $Pd_1 = \frac{1}{p}$.*

Proof: (sketch)

1. $Pd_0 = \frac{1}{p}$
   For an arbitrary message l and an arbitrary tag value $W \in H_p$, the number of keys e'_{IJ} for which $e'_{IJ}(l) = W$ is p. This is true because $e'_{IJ}(l) = g^i g^{jl} = g^w \pmod{P}$ where $I = H_p^{(i)}$, $J = H_p^{(j)}$, and $W = H_p^{(w)}$.
   That is the keys $e'_{IJ}(l)$ have to satisfy
$$i + jl = w \pmod{p}, \quad i, j \in Z_p$$
   This means that $Pd_0 = \frac{1}{p}$
2. To prove $Pd_1 = \frac{1}{p}$ using a similar type of argument and for an arbitrary pair of messages l and l', we must find
$$\max_{w,w'} |\{e'_{ij} : e'_{ij}(l) = w, e'_{ij}(l') = w'\}|$$

That is we need to find the number of (i, j) that satisfy

$$i + jl = w \pmod p \qquad \text{and} \qquad i + jl' = w' \pmod p$$

That is we need to find the number of js that satisfy

$$j(l - l') = w - w' \pmod p$$

This is equal to 1 as $j = (l - l')^{-1}(w - w') \pmod p$. So for any chosen i exactly one j can be found and so the number of (i, j) pairs is p which gives $Pd_1 = \frac{1}{p}$.

□

A FSS scheme based on the above families of A-codes

Our construction is based on $A(M, T, E)$, $A(M, T', E')$, and the bundling hash function defined above.

Model
There is a polynomially bounded sender $\tilde{\mathbf{S}}$, a polynomially bounded receiver $\tilde{\mathbf{R}}$ and an enemy $\tilde{\mathbf{E}}$ with unlimited computational power. The A-codes are public. We follow the second model of FSS (as in [13]) which does not require a trusted dealer. This scheme can be easily modified to the first model of FSS (as in [14, 3]) by replacing the role of $\tilde{\mathbf{R}}$ with a trusted dealer in the prekey generation phase.

Prekey Generation
$\tilde{\mathbf{R}}$ chooses two prime numbers p and q, $p < q < 2p$, and computes $n = pq$ and $P = 2pq + 1$, where P is also prime. If P is not a prime, $\tilde{\mathbf{R}}$ has to choose another set of p and q such that P is prime. He also chooses an element g of Z_P^* with $ord_P(g) = p$. Finally, he publishes n, g and P, and keeps p and q secret.

Key Generation
$\tilde{\mathbf{S}}$ chooses a secret key (i, j), and publishes his public key (γ_1, γ_2), where

$$\gamma_1 = g^i \pmod P$$

$$\gamma_2 = g^j \pmod P$$

Signing a Message ℓ
To sign a message $\ell \in Z_n$, $\tilde{\mathbf{S}}$ computes

$$t = i + j\ell \pmod n$$

where t denotes the signature or the tag of ℓ. The signed message is (ℓ, t).

Testing a Signature
(ℓ, t) passes the verification test if

$$\gamma_1 \gamma_2^{\ell} \overset{?}{=} g^t \pmod P$$

holds.

Proof of Forgery

If there is a forged signature t' that also passes the verification test, the presumed signer can prove that he has obtained a collision by showing his own signature t together with t'. That is,

$$\begin{aligned} g^t &= g^{t'} \pmod{P} \\ t &= t' \pmod{p} \\ t-t' &= kp,\ k \in Z \end{aligned}$$

Then, the presumed signer can find the factorisation of n by calculating $gcd(t-t',n)$ which is p.

Security Proof

The above construction conforms with the general construction described in Section 3 and to prove security of the scheme we need to find ϵ, ϵ' and $\max_{e'} \epsilon(e')$.

Theorem 4.3. *In the above construction* $\epsilon = 1/p$, $\epsilon' = 1/p$ *and* $\max_{e'} \epsilon(e') = 1/q$.

Proof. Using Theorems 4.1 and 4.2 we have $\epsilon = \epsilon' = 1/p$. To show $\max_{e'} \epsilon(e') = 1/p$, we need to find $Pd_0(e')$ and $Pd_1(e')$ for $A(M,T,E)$ restricted to $E(e') = \{e \in E : h(e) = e'\}$. Let e' be labelled by (I,J) where $I = H_p^{(i)}$ and $J = H_p^{(j)}$; that is $e'(l) = g^{(i+jl)}$. Firstly we note that $|E(e')| = q^2$. This is because the encoding rule labelled by $((i+mp),(j+np))$, where $m,n \in \{0,1,\cdots q-1\}$ are mapped to the same e'. Next For an arbitrary $t \in T$ and a message $l \in M$, we need to find

$$|\{e \in E(e') : e(l) = t\}|$$

That is find the number of solutions to

$$(i+mp)+(j+np)l = t \pmod{p}q$$

where $i + jl = t' \pmod{pq}$. This is equivalent to finding the number of solution to $(m+nl)p = t-t' \pmod{pq}$, or the number of solution to $m+nl = t-t' \pmod{q}$. Now for any l and an arbitrary n we can find a unique m that satisfies this equation. This is true because we have $nl = u \pmod{q}$ and so by choosing $m = t-t'-u$ the equation is satisfied. So we have $|\{e \in E(e') : e(l) = t\}| = q$ and $Pd_0(e') = 1/q$.

To find Pd_1, using a similar approach we need to find the number of solution to the following equations:

$$(i+mp)+(j+np)l = t \qquad \text{and} \qquad (i+mp)+(j+np)l' = t'$$

Let $i+jl = w \pmod{pq}$ and $i+jl' = w' \pmod{pq}$. This means that we must find the number of solutions to $m+nl = t-w \pmod{q}$ and $m+nl' = t'-w \pmod{q}$, or $n(l-l') = (u-u')+(w-w') \pmod{q}$, which is 1. Thus, $Pd_1(e') = 1/q$.

$\square$

From the above theorem we have $\max\{\epsilon, \epsilon', \max_{e'} \epsilon(e')\} = 1/p$. To construct a FSS with security parameter (σ,τ) we must choose $1/p \le 2^{-\sigma}$ or $p = \log_2 \sigma$ and $p < q < 2p$.

5. Conclusion

We proposed a general construction for FSS using authentication codes. We gave an example of this general construction and proved its security.

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*, pages 480–494, 1997.

[2] 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.

[3] I. B. Damgård, T. P. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10/3:163–194, 1997.

[4] W. Diffie and M. Hellman. New directions in cryptography. *IEEE IT*, 22:644–654, 1976.

[5] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17/2:281–308, 1988.

[6] L. Lamport. Constructing digital signatures from a one-way function. *PSRI International CSL-98*, 1979.

[7] A. K. Lenstra and E. R. Verheul. Selecting Cryptographic Key Sizes in Commercial Applications. *Price Waterhouse Coopers, CCE Quarterly Journal*, 3, 3-9, 1999. Full version appears in *http://www.cryptosavvy.com.*

[8] T. P. Pedersen and B. Pfitzmann. Fail-stop signatures. *SIAM Journal on Computing*, 26/2:291–330, 1997.

[9] B. Pfitzmann. Fail-stop signatures: Principles and applications. *Proc. Compsec '91, 8th world conference on computer security, audit and control*, pages 125–134, 1991.

[10] B. Pfitzmann. Fail-stop signatures without trees. *Hildesheimer Informatik-Berichte, Institut für Informatik*, 16/94, 1994.

[11] B. Pfitzmann. *Digital Signature Schemes – General Framework and Fail-Stop Signatures.* Lecture Notes in Computer Science 1100, Springer-Verlag, 1996.

[12] B. Pfitzmann and M. Waidner. Formal aspects of fail-stop signatures. *Interner Bericht, Fakultät für Informatik*, 22/90, 1990.

[13] 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*, pages 15–30, 1993.

[14] E. van Heyst and T. Pedersen. How to make efficient fail-stop signatures. *Advances in Cryptology - Eurocrypt '92*, pages 337–346, 1992.

[15] 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.

Centre for Computer Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, AUSTRALIA
E-mail address: {`rei, s05`}`@uow.edu.au`