On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures

Ivan B. Damgård *, Torben P. Pedersen ** and Birgit Pfitzmann ***

Abstract. We show that the existence of a statistically hiding bit commitment scheme with non-interactive opening and public verification implies the existence of fail-stop signatures. Therefore such signatures can now be based on any one-way permutation – the weakest assumption known to be sufficient for fail-stop signatures. We also show that genuinely practical fail-stop signatures follow from the existence of any collision-intractable hash function. A similar idea is used to improve a commitment scheme of Naor and Yung, so that one can commit to several bits with amortized O(1) bits of communication per bit committed to.

Conversely, we show that any fail-stop signature scheme with a property we call the *almost unique secret key property* can be transformed into a statistically hiding bit commitment scheme. All previously known fail-stop signature schemes have this property. We even obtain an equivalence since we can modify the construction of fail-stop signatures from bit commitments such that it has this property.

1 Introduction

In this section, we introduce the two main actors on the scene, fail-stop signatures (FSS) and statistically hiding bit commitments.

Fail-stop signatures were introduced in [16]. Further constructions appear in [13, 14, 5, 6]. A formal definition of the concept and a survey of the recent most efficient schemes will appear in [12].

Before going into the properties of FSS schemes, let us discuss some aspects of ordinary digital signatures: In an application of such signatures, what should happen if someone shows up with a message and a valid looking signature from user A, but A claims that she never signed the message? Suppose the signature scheme is based on a computational problem, P, which everybody accepts cannot be solved in polynomial time. Based on this one could claim that it is not reasonable to assume that the system was broken by an enemy. So either A is

* Aahus University, Matematisk Institut, Ny Munkegade, DK-8000 Aarhus C; Denmark. e-mail: ivan@daimi.aau.dk.

** Aarhus University, Matematisk Institut, Ny Munkegade, DK-8000 Aarhus C, Denmark. e-mail: tppedersen@daimi.aau.dk. Supported in part by the Carlsberg Foundation.

*** Universität Hildesheim, Institut für Informatik, Marienburger Platz 22, D-31141 Hildesheim, Germany. e-mail: pfitzb@informatik.uni-hildesheim.de.

lying, or she must have stored her secret key insecurely, and should therefore be held responsible in either case.

However, this argument sweeps under the rug a very important point: we always have to choose particular instances of the problem for each user, and the discussion should actually refer to how hard this particular instance is to break. If we are using RSA, for example, we have to decide on a size of moduli to use. Even if we believe that factoring is not in polynomial time, this does not answer questions like: "are 512-bit moduli secure enough?". This is a question about the state of the art of practical factoring, and does not have much to do with its complexity theoretic status.

In a practical situation, it is often the case that individual users have only very limited computing power available. This of course limits the size of problem instance they can use, but not the amount of computing power that might be used to break those instances. In such a situation, depending on the practical circumstances, the possibility that A is not lying and someone broke her key, is perhaps not so unreasonable after all.

This raises a natural question: is it possible at all to distinguish between on one hand the case where A is lying or has leaked the secret key, and on the other hand the case where someone with a large (unexpected) amount of computing power has broken the system?

This is precisely what FSS schemes enable us to do. The crucial property that distinguishes FSS from ordinary digital signatures is that there are several possible secret keys corresponding to a given public key. Even an infinitely powerful enemy cannot guess from publicly available information which of the possible secret keys is known to the signer.

Since usage of different secret keys in general leads to different signatures, it is impossible for the enemy to predict which signature the signer would produce on a given message, if it has not already been signed.

Furthermore, from two different signatures on the same message, A can produce what is known as a *proof of forgery*. But if she has only the signature available that she would produce herself, it is not feasible for her to produce such a proof.

Thus if a powerful enemy tries to frame A and submits a message seemingly signed by A, with overwhelming probability the signature will not be the one A would produce herself, and A can therefore respond with a proof of forgery. On the other hand, A cannot falsely repudiate her own signature, if it has in fact not been forged, unless she herself breaks the computational assumption. (Thus even in this case, the proof of forgery correctly indicates that someone with unexpectedly large computing power has broken the scheme.)

In this paper, we show that there is an intimate connection between statistically hiding bit commitment schemes and FSS schemes. A bit commitment scheme is a protocol that party A can conduct with B to commit herself to a bit b without revealing to B (or anyone else) the value of b. At a later time, A can open the commitment and convince B about the value that was chosen originally, i.e., it is not feasible for A to open a commitment to reveal both $b = 0$

and $b = 1$. A commitment scheme is said to be *statistically hiding* if B gets only negligible Shannon-information about b prior to the opening of the commitment. Such bit commitment schemes are extremely important because their existence implies perfect or statistical zero-knowledge arguments for any problem in NP.

Concretely, we show how to construct FSS schemes from any statistically hiding bit commitment scheme with non-interactive opening and public verification (see below for details). This result is also contained more or less implicitly in [13]. Also the result was discussed informally, prior to the work on this paper, by Moti Yung and Birgit Pfitzmann. Our contribution in this respect is to somewhat simplify the construction and to identify the properties needed from the bit commitment scheme.

By the work of Naor et al. ([11]), this means that FSS schemes can be based on any one-way permutation. Before, FSS schemes were only known to follow from the existence of claw-free permutations.

We also show that any collision-intractable hash function can be used to build a secure FSS scheme. If the hash function is efficient, like MD4 [7] or SHA [15], say, then the resulting FSS scheme is practical.

Conversely, we show that any FSS scheme with a property we call the *almost unique secret key property*, can be transformed into a statistically hiding bit commitment scheme. This property means that it is infeasible for a signer to compute more than one significantly different secret key corresponding to her public key; see below for details. All previously known FSS schemes have this property. Finally, we show that the existence of FSS schemes with this property is in fact *equivalent* to the existence of statistically hiding bit commitments with non-interactive opening and public verification.

2 Definitions and Notation

2.1 Fail-Stop Signatures

For the results in this paper, it is sufficient to consider fail-stop signature schemes that allow just one message to be signed. Based on [12] (see also [13]), the definition of such schemes is now sketched.

A fail-stop signature scheme consists of five parts: a protocol for generating the keys, a method for signing, a predicate for verifying signatures (a signature satisfying this predicate is called *acceptable*), a method for constructing proofs of forgery and a predicate for verifying proofs of forgery (a proof satisfying this predicate is called *valid*). The methods for producing signatures and proofs of forgery take the secret key as input, and the public key is input to the computation of the two predicates.

Unlike usual digital signatures, the key generation is a two-party protocol, which is executed by the signer A and a center B trusted by the recipients. This is necessary to ensure that the signer does not generate a pair of keys for which she can prove her own signatures to be forgeries. A or B may reject in key generation, but if both parties are honest, this should only happen with negligible probability. We remark that one can always do without a key center (at the expense of efficiency), by letting every recipient play the role of B.

Obviously, these parts must satisfy that if the keys are generated correctly, then correct signatures and correct proofs of forgery are accepted by the corresponding verification methods. The more interesting parts of the definition are

- *Security for the recipient*: It is infeasible for a polynomially bounded signer to produce an acceptable signature and a valid proof that it is forged.
- *Security for the signer*: It is impossible for a forger with unlimited computing power to produce a signature that the signer cannot prove to be a forgery.

A fail-stop signature scheme usually has two security parameters: k for the security of the recipient and σ for the security of the signer.

To define the security of the recipient in more detail, we consider the following scenario involving the key center, B, and a possibly cheating signer, $\tilde{A}$: First $\tilde{A}$ and B generate a pair of keys, and then $\tilde{A}$ outputs a triple (m, s, pr).

Definition 2.1 *A fail-stop signature scheme is secure for the recipient with respect to the security parameter k if for all $c > 0$ and for all polynomially bounded signers, $\tilde{A}$, the following holds for sufficiently large k: The probability that s is an acceptable signature on m and pr is a valid proof of forgery is at most k^{-c}. This probability is over the random coins of B and $\tilde{A}$.*

In order to define the security of the signer, we consider a cheating center, $\tilde{B}$, possibly with unlimited computing power. As the signer must be secure even if the center cooperates with future recipients, it is sufficient that the center itself cannot construct forgeries that A cannot disavow. Consider the scenario where first A and $\tilde{B}$ execute the key generation protocol. This results in a secret key, sk, and a public key, pk. $\tilde{B}$'s view of this protocol is denoted by $view_{\tilde{B}}$ (random bits and all messages). Then $\tilde{B}$ outputs a pair (m_0, s_0), where s_0 should be an acceptable signature on m_0.

Let SK be the set of possible secret keys given $view_{\tilde{B}}$. SK is equipped with a probability distribution induced by the random coins used by A during the key generation. Then *Good* is defined as the set of pairs $(pk, view_{\tilde{B}})$ such that for all m_0, s_0 and with probability at least $1 - 2^{-\sigma}$ over the choices of possible secret keys sk in SK, A can prove that s_0 is a forgery using sk as the secret key.

Definition 2.2 *A fail-stop signature scheme is secure for the signer with respect to the security parameter σ, if the probability that A accepts the keys and $(pk, view_{\tilde{B}}) \notin Good$ is at most $2^{-\sigma}$. (The probability is over A's coins).*

Intuitively, this means that only with very small probability ($\leq 2^{-\sigma}$) can $\tilde{B}$ make A accept a pair of keys for which she has probability less than $1 - 2^{-\sigma}$ of proving forgeries.

A complete definition also has to take into account chosen message attacks. However, our construction of commitments from FSS does not need security against such attacks, and for the FSS schemes we construct, it is easy to see that such attacks make no difference. Hence, we stick to this somewhat simpler definition.

Intuitively, these definitions imply that a cheating signer cannot compute just any secret key that is possible given the public key. If she could, she could prove her own signatures to be forgeries by signing using one secret key and using a different key in the proof. All fail-stop signature schemes in previous literature have an idealized version of this property: No matter how the (polynomially bounded) signer executes the key generation, she cannot compute two different secret keys that are both possible given the public key. We call this the *unique secret key property*. In the following, we use a relaxed version of it, the *almost unique secret key property*: Although the signer might be able to find more than one secret key fitting a public key, she cannot not find *significantly different ones*. Keys are "not significantly different" if they lead to equal signatures. This can be formalized by introducing a polynomial-time computable mapping κ on the secret keys with the intuitive meaning that $\kappa(sk)$ is the part of sk that makes a difference in the signatures.

Definition 2.3 *A fail-stop signature scheme has the almost unique secret key property if there are a polynomial-time computable predicate $Fits$ and a polynomial time computable mapping κ with the following properties:*

- *If the signer follows the key generation protocol, the resulting secret and public key, sk and pk, always fulfil $Fits(sk, pk) = 1$.*
- *No probabilistic poly-time bounded signer can execute the key generation protocol with the honest key center and compute sk_1, sk_2 such that $\kappa(sk_1) \neq \kappa(sk_2)$ and $Fits(sk_1, pk) = Fits(sk_2, pk) = 1$ with more than superpolynomially small probability.*
- *If sk_1 and sk_2 satisfy $Fits(sk_1, pk) = Fits(sk_2, pk) = 1$ and $\kappa(sk_1) = \kappa(sk_2)$, then for any message, the signature produced with sk_1 equals the one produced with sk_2.*

For a concrete FSS scheme, there will typically exist a function that computes the public key from a secret key. Then $Fits$ can be constructed from this function. Furthermore, note that if κ is the identity, the third property is no restriction, and one just obtains the unique secret key property.

All known schemes (see [13, 14, 5, 6]) have the almost unique secret key property, although one can easily construct artificial schemes without it.

2.2 Bit Commitments

We define a bit commitment scheme as a pair of two-party protocols, namely the *commit* and the *reveal* protocol. They take place between parties A and B, where A is the party committing herself. The participants are modeled in the standard way as interactive probabilistic Turing machines. As before, the *view* of a participant is the bit string consisting of his own coinflips concatenated by all messages sent in the protocol. $\tilde{X}$ will denote any machine playing the role of X in the protocol.

For the commit protocol, A gets as input a bit b. We assume that B knows some a priori information about b, such that $b = 0$ with probability δ, where δ may be different from 1/2. In addition both parties have access to a security parameter k. The concatenation of all messages sent in the commit protocol is called *the commitment*. In some concrete schemes, it makes sense to define the commitment as a subset or a function of the messages. We have chosen our definition of a commitment for simplicity. A or B may reject in the commit protocol, but if both parties are honest, this should only happen with negligible probability.

For the reveal protocol, A gets as input her view of the commit protocol, while B gets the commitment as input. At the end of the reveal protocol, B outputs *reject*, *accept* 0 or *accept* 1. The intuitive meaning is that either B has detected cheating by A, or he accepts that A has opened the commitment to reveal either 0 or 1.

We will only consider commitment schemes with non-interactive opening, i.e., where the reveal protocol consists of A sending one message to B.

A statistically hiding bit commitment scheme must satisfy two properties:

- *Security Property:* For any $\tilde{B}$, let *bias* denote $\tilde{B}$'s advantage in guessing b given $\tilde{B}$'s view v of the commit protocol, i.e., $bias = |\delta - Prob[b = 0 \mid v]|$. Then the expected value of *bias* is at most 2^{-k}. The probabilities are taken over the coinflips of A.
- *Binding Property:* Let $\tilde{A}$ be any polynomially bounded machine that executes the commit protocol with B, and then outputs two strings s_0, s_1.
  Let $p(\tilde{A}, k)$ be the probability that B outputs *accept* b on input s_b in the reveal protocol, for both $b = 0$ and $b = 1$. The probability is taken over the coinflips of B and $\tilde{A}$. Then $p(\tilde{A}, k)$ is superpolynomially small as a function of k.

We require an exponential decrease of the bias in the security property. However, this is not a significant restriction, since: standard "XOR-ing" techniques can be used to improve weaker schemes such that they satisfy the definition. Moreover, most practical examples known in fact have a bias of 0.

Note that we have built into the model two properties that the bit commitment scheme must satisfy in order for the result in the next section to work: first non-interactive opening must be possible, as mentioned; secondly B must be able to verify the opening based on the commitment only. This means that anyone who trusts that a given commitment is the result of a conversation with B can verify the opening without knowing B's coinflips. Hence the term *public verification*. This property is necessary in the construction of a FSS scheme to ensure that everybody can verify signatures.

3 Fail-Stop Signatures from Bit Commitments and Hash Functions

In this section, we assume that we are given a statistically hiding bit commitment scheme as defined above, and will use this to build an FSS scheme. The basic idea is very similar to Lamport and Diffie's one-time signatures.

We will only show how to sign a 1-bit message – this easily generalizes to any number of bits. Recall that an FSS scheme uses two security parameters, k and σ.

Key generation
In this phase, the key center B and user A execute 4σ instances of the commit protocol, which is executed with security parameter k' such that $k' \geq k$ and $k' \geq 2\sigma + 4$. For each instance, A chooses randomly and uniformly the bit to commit to. The resulting commitments are organized in 2σ pairs called $(C_{0,i}, C_{1,i}),\ i = 1, \ldots, 2\sigma$. The public key is the set of commitments, while the secret key is the set of (4σ) strings known by A that will open the commitments. A stops and rejects the keys, if she detects cheating during any of the commit protocols.

Signing
The signature on a bit b consists of the 2σ strings that A would send to open the commitments $C_{b,i},\ i = 1, \ldots, 2\sigma$.

Verification
To verify the signature on a bit b, one verifies that the 2σ strings in the signature open correctly the commitments $C_{b,i},\ i = 1, \ldots, 2\sigma$.

Proof of forgery
Given an acceptable signature S on a bit b, A generates her own signature on b. For $i = 1, \ldots, 2\sigma$, she tries to find an i for which the i'th bit opened in S is different from the i'th bit opened in her own signature. If such an i is found, she outputs i and the two strings used to open this commitment. If not, she fails to generate a proof of forgery.

Validating proof of forgery
A triple, (i, s_1, s_2) proves that S is a forged signature on b, if s_1 is the i'th string in S $(1 \leq i \leq 2\sigma)$, $s_1 \neq s_2$, and s_1 and s_2 can be used to open $C_{b,i}$ to reveal different bits.

Theorem 3.1 *The signature scheme outlined above based on a statistically hiding bit commitment scheme with public verification and non-interactive opening is a secure fail-stop signature scheme.*

Proof sketch: We first prove security for the signer. Let Acc denote the event A accepts the keys. As the event $G = Good$, we take the event that B cannot guess any bit committed to with probability better than 5/8. For a single commitment, by the security property and Markov's rule, the probability that B's guess is better than 5/8, is at most $8 \cdot 2^{-k'} \leq 2^{-2\sigma-1}$. Therefore we get that

$$Prob[Acc, \neg G] \leq 1 - (1 - 2^{-2\sigma-1})^{4\sigma} < 4\sigma 2^{-2\sigma-1} \leq 2^{-\sigma}.$$

This shows the first part of security for the signer. Next, assume G occurs. Then, even an infinitely powerful enemy cannot predict in which way A will open any commitment with probability better than $5/8$. To predict A's signature, one must guess the contents of 2σ commitments, which can be done with probability at most $(5/8)^{2\sigma} \leq 2^{-\sigma}$. Therefore the probability that the algorithm for generating a proof of forgery fails when given a false signature (i.e., a signature generated by anyone else than A) is less than $2^{-\sigma}$. This implies security for the signer.

Note that it does not help B if he first gets a signature from A (e.g., if A signs the bit 1 and B wants to forge the signature on 0), because the commitments are chosen independently.

Furthermore, it is clear that any algorithm that would allow A to generate a proof of forgery by herself would also allow her to break the binding property of the commitment scheme. This implies security for the recipient. ∎

Corollary 3.2 *If one-way permutations exist, then there exists a secure FSS scheme.*

Proof sketch: In [11], a statistically (in fact perfectly) hiding bit commitment scheme is constructed from any one-way permutation. It is easy to check that this commitment scheme has the properties of public verification and non-interactive opening. ∎

It is clear that the signature scheme we just constructed is a one-time signature scheme, and therefore not very efficient. If the commitment scheme we use needs interaction for every new commitment, however, there does not seem to be a way around this.

However, with a (perhaps) stronger assumption we can do much better, namely the assumption that collision-intractable (collision-free) hash functions exist.

Assume we have a family, H, of collision-intractable hash functions, such that functions in the family map $(k + 2\sigma + 1)$-bit inputs to k-bit outputs. Functions in the family can be computed easily and can be efficiently selected at random, but the probability that a poly-time bounded enemy can find collisions for a member of H is superpolynomially small in k.

Note that we can build such a family with the right input length from any collision-intractable family by fixing some input bits if the input length is too large, and using the iterative construction of [4] if inputs are too short. Now, we have the following observation:

Lemma 3.3 *Let h be any function from $k + 2\sigma + 1$ bits to k bits. Then, when x is uniformly chosen, the probability that the preimage of $h(x)$ has size at least 2^σ is at least $1 - 2^{-\sigma-1}$.*

Proof: Let the degree of a point in the image of h be the size of its preimage under h. Since h maps into the set of k-bit strings, at most $2^k \cdot 2^\sigma$ elements can be preimages of elements of degree $\leq 2^\sigma$. Hence a uniformly chosen x is such a preimage with probability at most $2^{k+\sigma}/2^{k+2\sigma+1}$. ∎

We can use this result to build a simple FSS scheme along the lines of the one described in detail above: To generate keys, B chooses a hash function h from the family, sends it to A, and A chooses two preimages (her secret key) x_0, x_1 and sends the public key $h(x_0), h(x_1)$ to B. The signature on bit b will be x_b. A proof of forgery for a signature on bit b will be two different preimages of $h(x_b)$.

This scheme is secure for the recipient by the collision-intractability of H, and secure for the signer by Lemma 3.3: the event *Good* is that both parts of the public key have preimages of size at least 2^σ. (If more than one bit is signed, the probabilities that one of the preimages is too small accumulate, but this can be countered by letting the size of the inputs to the hash function grow logarithmically with the number of such public keys used.)

This is still just a one-time signature scheme. But since we have a collision-intractable hash function, we can use $2k$ pairs from the public key to authenticate the hashed image of any number of new pairs, and thus make an arbitrary number of signatures in a tree-like structure, in the style of Merkle's signature schemes [8, 9]. We can get the public key even shorter by hashing the original pairs down to k bits. Similarly, messages of arbitrary length can be also signed using only k pairs by hashing them first.

False signatures can then also be generated by finding new messages colliding under h with already signed ones. But this is not a problem for generating proofs of forgery: the signer can show the collision as a proof (which by collision-intractability she could not generate herself).

Note that, since the function h can be the same for all signers, it makes sense not to count the description of h as a part of the public key. Thus we have shown:

Theorem 3.4 *If collision-intractable hash functions exist, there exists a secure fail-stop signature scheme where an arbitrary number of messages can be signed, and the length of the public key is just the security parameter k.*

Since extremely efficient hash functions exist in practical applications (MD4, SHA, etc..), this shows that really practical FSS schemes can be constructed based on conventional cryptography only.

4 Efficient Statistically Hiding Commitments

Naor and Yung [10] have shown that a statistically hiding bit commitment scheme can be built from collision-intractable hash functions. This scheme needs interaction only in an initialization phase, after which both committing and opening are non-interactive.

We now sketch how to modify the Naor-Yung scheme to get more efficient commitments, where several bits can be committed to at once. The amortized number of bits of communication per bit committed to is only O(1). Our scheme makes use of families of universal hash functions [3]. These functions are interesting because they emulate some properties of random functions, although they have much shorter descriptions, and can therefore be efficiently used in protocols. The standard example of a family of universal hash functions from n-bit strings to $i \leq n$-bit strings are the functions that map x to $ax + b|_i$, where $|_i$ means that we take only the most significant i bits, and where $a, b \in GF(2^n)$. Thus each member of the family is characterized by a choice of a, b. This family is 2-universal, which means that for any 2 fixed inputs x_1, x_2, the images $f(x_1), f(x_2)$ are uniformly and independently distributed i-bit strings, when f is uniformly chosen from the family.

Let t denote the number of bits to be committed to and k the security parameter of the scheme, and let H be a family of collision-intractable hash functions constructed such that the input length is $2k + t$ whenever the output length is k. Consider the following commitment scheme:

Initialization Phase
B chooses at random a function $h \in H$ with output length k bits. He sends h to A.

Commit Protocol
A chooses at random a $(2k + t)$-bit string x, and a 2-universal hash function f from $2k+t$ bits to t bits. Let $\bar{b} = b_1, \ldots, b_t$ be the t-bit string A wants to commit to. She then sends f, $h(x)$ and the bitwise XOR $C = \bar{b} \oplus f(x)$ to B.

Reveal Protocol

1. A sends $\bar{b}$ and x to B.
2. B checks that h maps x to $h(x)$, and compares C to $\bar{b} \oplus f(x)$. If OK, he accepts the opening, otherwise he rejects.

A formal proof of security for this commitment scheme would require a generalization of the definition in Section 2.2 to commitments to many bits. We have omitted this for simplicity, and therefore only sketch the proof below.

Theorem 4.1 *The scheme described above is a statistically hiding commitment scheme, under the assumption that H is a family of collision-intractable hash functions. It allows commitment to t bits by a commitment of size $5k + 3t$ bits.*

Proof sketch: The size of commitments is clear from the description above.

The binding property is trivial from the collision-intractability of H. For the security property, the privacy amplification theorem of [1] (see also [2]) says that, over the choice of x and f, B's expected information about $f(x)$ (and therefore about $\bar{b}$) given by knowledge of f, h, and $h(x)$ is at most $2^{-k}/\ln 2$. ■

5 Bit Commitments from Fail-Stop Signatures

The main idea in our construction of bit commitments from FSS schemes is to use the key generation protocol between user A and key center B as the commit protocol, and to think of the resulting public key as the commitment and the secret key as the string that can open the commitment.

If the FSS scheme has the almost unique secret key property, it is obvious that A is committed to any value that can be computed from $\kappa(sk)$, where sk is the secret key. There are two major difficulties, however: First, the distribution of the secret key held by A given the public key is not necessarily uniform. So we need a way to assign a value to the secret key known by A in such a way that B has essentially no information about it, given the public key. This is done by using universal hash functions [3] and the extended privacy amplification result of [1]. Secondly, the definition of FSS schemes somewhat counterintuitively allows the key generation to lead to a secret key that can be guessed by a dishonest key center. This may happen for keys with the strange property that the signer can prove that (her own) signatures made with this secret key are forgeries. Such keys are not necessarily unlikely if the key center is dishonest. Hence we must provide a way for the signer (now the committer) to exclude these keys.

We now give a more detailed description of the construction:

COMMIT PROTOCOL

1. A and B execute the key generation protocol of the FSS scheme with security parameters (k, σ), where $\sigma = 4k + 4$, and k equals the security parameter for the bit commitment scheme we are building. Here B plays the role of the key center. If A or B reject in the key generation, the commit protocol stops. Otherwise let sk be the resulting secret key and pk the public key.
2. A signs the message, "0" (consisting of one 0-bit) using sk. She runs the algorithm for generating proofs of forgery on the resulting signature. If this results in a proof of forgery, she stops. Otherwise she continues.
3. A chooses and sends to B a random 2-universal hash function h with a 1-bit image.
4. Let b be the bit A wants to commit to. Then A sends $c = h(\kappa(sk)) \oplus b$ to B.

OPENING

1. A sends b and sk to B.
2. B verifies the secret key, by checking that $Fits(sk, pk) = 1$. He then compares b with $c \oplus h(\kappa(sk))$. If they are equal, he outputs *accept* b, if not, he outputs *reject*.

Theorem 5.1 *If the above construction is based on a secure FSS scheme with the almost unique secret key property, the result is a statistically hiding bit commitment scheme (with non-interactive opening and public verification).*

Proof: First note that the possibility of stopping in Step 2 does not prevent an honest A and B from completing the protocol: security for the recipient implies that the scheme almost never stops in Step 2 if A and B are honest.

The binding property is clear from the almost unique secret key property of the FSS scheme: if the committer could open the commitment in two different ways, she would know two secret keys satisfying the predicate $Fits$ and with different κ-images.

For the security property, we need the following notation: Let Acc be the event that A accepts the key generation, i.e., does not stop in Step 1, and let U be the event that A does not stop in Step 1 or 2. Finally let G be the event that the public key produced, pk, and $view_{\tilde{B}}$ are in the set $Good$, as defined before Definition 2.2.

The extended privacy amplification theorem from [1] deals with collision entropies, instead of Shannon entropies. The collision entropy, or Renyi entropy, of a distribution is defined as minus the logarithm base 2 of the sum of the squared probabilities. For a binary distribution, like that of $h(\kappa(sk))$, with probabilities p and $1-p$, the collision entropy is

$$R(p) = -\log_2(p^2 + (1-p)^2).$$

This is a value between 0 and 1, like the Shannon entropy. It is therefore natural to define the *collision information* to be $1 - R$.

Let $I_{\tilde{B}}$ denote the collision information obtained by $\tilde{B}$ about $h(\kappa(sk))$ during the commit protocol, and let $E_{\tilde{B}}$ be its expected value, taken over the random choices of A. For an event X, $E_{\tilde{B}}(X)$ denotes the expected information given that X occurs. Then we have

$$\begin{aligned} E_{\tilde{B}} &= Prob[\neg U]E_{\tilde{B}}(\neg U) + Prob[U,G]E_{\tilde{B}}(U,G) + Prob[U,\neg G]E_{\tilde{B}}(U,\neg G) \\ &\leq 0 + Prob[U,G]E_{\tilde{B}}(U,G) + Prob[Acc,\neg G] \\ &\leq Prob[U,G]E_{\tilde{B}}(U,G) + 2^{-\sigma} \end{aligned}$$

by the security for the signer, and since A does not reveal anything at all if the commit protocol is aborted in Step 1 or 2.

The rest of the proof proceeds in 3 parts: We first show that in most cases, the best guess at the significant part of the secret key from the point of view of $\tilde{B}$ still has a rather small probability of being correct. Secondly, we derive with the extended privacy amplification theorem that in most cases, an enemy has very little collision information about $h(\kappa(sk))$. Finally, we derive an upper bound on the advantage an enemy has in guessing the content of the commitment.

Part 1 Let SK be the random variable denoting the secret key of A, and let sk_{max} denote a secret key such that $\kappa(sk_{max})$ has maximal probability given $v := (pk, view_{\tilde{B}})$, U and G. We now show that *on average* over the possible v's and given U and G, this maximal probability is upper bounded:

$$Prob[\kappa(SK) = \kappa(sk_{max}) \mid U, G] \leq 2^{-\sigma} Prob[U,G]^{-1}. \qquad (*)$$

For this, it is sufficient to show that

$$Prob[\kappa(SK) = \kappa(sk_{max}), U, G] \leq 2^{-\sigma}.$$

To do this, we consider the following attack by B^* on the FSS scheme.

1. B^* executes the key generation protocol with A in the same way as $\tilde{B}$ did.
2. B^* finds sk_{max} and uses it to make a signature on the message "0".

Let F denote the event that A fails to prove this forgery. Note that the distribution of the keys after Step 1 of this attack and of the commit protocol are equal. Furthermore, $U \subset Acc$, and whenever Acc and $\kappa(SK) = \kappa(sk_{max})$ occur, U implies F by definition of the almost unique secret key property. This gives us

$$\begin{aligned} Prob[\kappa(SK) = \kappa(sk_{max}), U, G] &= Prob[\kappa(SK) = \kappa(sk_{max}), U, G, Acc] \\ &\leq Prob[F, \kappa(SK) = \kappa(sk_{max}), G, Acc] \\ &= Prob[F, Acc, \kappa(SK) = \kappa(sk_{max}) \mid G] Prob[G] \\ &\leq Prob[F \mid G] \\ &\leq 2^{-\sigma}. \end{aligned}$$

The final inequality follows from the security for the signer of the FSS scheme. This finishes the proof of (*).

Now let V be a random variable denoting $v = (pk, view_{\tilde{B}})$ and M the set of cases where the probability of the best guess is much larger than on average, and also the event that such a case occurs:

$$M := \{v \mid Prob[\kappa(SK) = \kappa(sk_{max}) \mid G, U, V = v] \geq Prob[U, G]^{-1} 2^{-\sigma/2}\}.$$

By Markov's rule, the average inequality $(*)$ implies that

$$Prob[M \mid U, G] \leq 2^{-\sigma/2}.$$

We split the expected information according to whether M occurs or not.

$$\begin{aligned} E_{\tilde{B}}(U, G) &\leq Prob[M \mid U, G] + Prob[\neg M \mid U, G] E_{\tilde{B}}(U, G, \neg M) \\ &\leq 2^{-\sigma/2} + \sum_{v \notin M} Prob[V = v \mid U, G] E_{\tilde{B}}(U, G, V = v). \end{aligned}$$

Part 2 Whenever $v \notin M$, the extended privacy amplification lemma, Lemma 5.2 below, immediately implies that the information $E_{\tilde{B}}(U, G, V = v)$ is small:

$$E_{\tilde{B}}(U, G, V = v) \leq Prob[U, G]^{-1} 2^{-\sigma/2} \frac{2}{\ln 2} < Prob[U, G]^{-1} 2^{2-\sigma/2}.$$

Substituting this into the two equations above where $E_{\tilde{B}}$ was partitioned gives

$$\begin{aligned} E_{\tilde{B}}(U, G) &< 2^{-\sigma/2} + \sum_{v \notin M} Prob[V = v \mid U, G] Prob[U, G]^{-1} 2^{2-\sigma/2} \\ &\leq 2^{-\sigma/2} + Prob[U, G]^{-1} 2^{2-\sigma/2} \end{aligned}$$

and thus

$$E_{\tilde{B}} \leq Prob[U, G] 2^{-\sigma/2} + 2^{2-\sigma/2} + 2^{-\sigma} < 2^{3-\sigma/2}.$$

Part 3 Let $Bias$ be the random variable denoting $\tilde{B}$'s advantage β in guessing $h(\kappa(sk))$. (This also bounds the advantage in guessing b.) ¿From the definition of the collision entropy for binary distributions one sees

$$R(1/2+\beta) = -\log_2(\frac{1}{2}+2\beta^2) = 1-\log_2(1+4\beta^2) \leq 1-4\beta^2$$

for $|\beta| \leq \frac{1}{2}$, i.e., $4\beta^2 \leq 1$. This implies

$$\beta \leq \frac{1}{2}\sqrt{1-R(1/2+\beta)}.$$

Thus we have shown the following pointwise inequality between the random variables $Bias$ and the collision information $I_{\tilde{B}}$:

$$Bias \leq \frac{1}{2}\sqrt{I_{\tilde{B}}}.$$

Applying the general formula $E(X) \leq \sqrt{E(X^2)}$ to $X = \sqrt{I_{\tilde{B}}}$ yields

$$E(Bias) \leq \frac{1}{2}E(\sqrt{I_{\tilde{B}}}) \leq \frac{1}{2}\sqrt{E(I_{\tilde{B}})} = \frac{1}{2}\sqrt{E_{\tilde{B}}} < 2^{1-\frac{\sigma}{4}}.$$

In the last inequality, the result of Part 2 was used.

As the security parameter of the FSS scheme, σ, equals $4k+4$, this shows that the commitment scheme has the security property. ■

Remark: From the proof of the security property, it is clear that we do not need to hash all the way down to a 1-bit value to wipe out the enemy's information. Therefore we can commit to more than one bit in one commitment.

Lemma 5.2 *Let S be a random variable with a given distribution $\{p_i \mid i = 1, 2, \ldots, 2^n\}$ on the set of n-bit strings. If there is an $\alpha > 0$ such that $p_i \leq \alpha$ for each i, and a random 2-universal hash function mapping n bits to 1 bit is chosen, then the expected collision information about the image $h(S)$ is at most $\alpha\frac{2}{\ln 2}$.*

Proof: Let $N = 2^n$. Then

$$\sum_{i=1}^{N} p_i^2 \leq \sum_{i=1}^{N} p_i\alpha = \alpha.$$

Hence the collision entropy R of the given distribution is

$$R \geq -\log_2(\alpha).$$

Theorem 5 of [1] shows that if an unbounded enemy knows at most l bits of collision information about an n-bit string (defined as $n-R$), and if the string is hashed down to $n-l-s$ bits, the enemy's expected collision information about the result is at most $2^{-s}/\ln(2)$ bits. In our case, we hash down to 1 bit, and thus $s = R-1 \geq -\log_2(\alpha)-1$. Hence, the enemy's expected collision information E about $h(S)$ is

$$E \leq \alpha \cdot \frac{2}{\ln(2)}.$$

■

6 Equivalence

The FSS scheme constructed from bit commitments in Section 3 does not necessarily have a unique secret key property: for example, more than one bit string may be acceptable as opening a commitment as a 1. In the following, we modify the scheme so that it has at least the almost unique secret key property. Together with Theorem 5.1, this yields the equivalence between FSS schemes with the almost unique secret key property and statistically hiding bit commitment schemes with non-interactive opening and public verification. The only changes to the protocol are:

- In key generation, A makes an additional commitment to every bit in the strings that will open the commitments $C_{b,i}$. These *secondary commitments* belong to the public key, and the strings that open them belong to the secret key.
- The security parameter σ is increased a little (since the secondary commitments may give a small amount of extra information).

It is clear that this scheme is still secure. For the almost unique secret key property, $Fits(sk, pk)$ is defined to mean that the strings in sk open all the commitments of pk correctly, and the significant part, $\kappa(sk)$, of sk, consists of those strings that open the original commitments. Clearly, finding two secret keys fitting the same public key, but with different κ-images, would mean opening at least one secondary commitment in both ways. Moreover, two secret keys sk_1, sk_2 with $\kappa(sk_1) = \kappa(sk_2)$ obviously lead to the same signatures.

7 Conclusion

We have shown that the existence of FSS schemes with the almost unique secret key property is equivalent to the existence of bit commitment schemes with non-interactive opening. In addition, we have shown how to construct efficient FSS schemes from any collision-intractable hash function.

Acknowledgement

It is a pleasure to thank *Moti Yung* for interesting discussions about all sorts of statistically hiding schemes, and in particular, one that lead to one of the independent discoveries of the construction of fail-stop signature schemes from bit commitments. We also thank *Michael Waidner* for letting us extend a joint result that has never really been published.

References

1. C. H. Bennett, G. Brassard, C. Crépeau, U. Maurer: *Privacy Amplification Against Probabilistic Information.* In preparation.

2. C. H. Bennett, G. Brassard, J.-M. Robert: *Privacy Amplification by Public Discussion.* SIAM Journal on Computing, vol 17, no. 2, 1988, pp. 210–229.
3. J. L. Carter, M. N. Wegman: *Universal Classes of Hash Functions.* Journal of Computer and System Sciences 18, 1979, pp. 143–154.
4. I. B. Damgård: *A Design Principle for Hash Functions.* Proceedings of Crypto'89, LNCS 435, pp. 416–427, 1990.
5. E. van Heyst, T. P. Pedersen: *How to Make Efficient Fail-Stop Signatures.* Presented at Eurocrypt'92, Balatonfüred, Hungary, 1992.
6. E. van Heyst, T. P. Pedersen, B. Pfitzmann: *New Constructions of Fail-Stop Signatures and Lower Bounds.* Presented at Crypto'92, Santa Barbara, 1992.
7. R.Rivest: *The MD4 message-digest algorithm*, Proc. of Crypto 90.
8. R. C. Merkle: *Protocols for Public Key Cryptosystems.* In: Secure Communications and Asymmetric Cryptosystems, AAAS Selected Symposium 69, G. J. Simmons (ed.); Westview Press, Boulder 1982, pp. 73–104.
9. R. C. Merkle: *A digital signature based on a conventional encryption function.* Proceedings of Crypto'87, LNCS 293, Springer-Verlag, Berlin 1988, pp. 369–378.
10. M. Naor, M. Yung: *Universal One-Way Hash Functions and their Cryptographic Applications.* Proceedings of 21^{st} STOC, pp. 33–43, 1989.
11. M. Naor, R. Ostrovsky, R. Venkatesan, M. Yung: *Perfect Zero-Knowledge Arguments for NP Can Be Based on General Complexity Assumptions.* Presented at Crypto'92, Santa Barbara, 1992.
12. T. P. Pedersen, B. Pfitzmann: *Fail-Stop Signatures.* Manuscript, February 1993.
13. B. Pfitzmann, M. Waidner: *Formal Aspects of Fail-Stop Signatures.* Internal report 22/90, Fakultät für Informatik, Universität Karlsruhe.
14. B. Pfitzmann, M. Waidner: *Fail-Stop Signatures and their Application.* Securicom 91, Paris, pp. 145 – 160.
15. *Specifications for a Secure Hash Standard*, Federal Information Processing Standards Publication YY, 1992.
16. M. Waidner, B. Pfitzmann: *The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability.* Proceedings of Eurocrypt'89, LNCS 434, page 690, 1990.

This article was processed using the LTEX macro package with LLNCS style