To appear in Proc. Crypto '92, Santa Barbara, 16.-20.8.1992, LNCS, Springer-Verlag

New Constructions of Fail-Stop Signatures and Lower Bounds

(Extended Abstract)

Eugène van Heijst[1], Torben Pryds Pedersen[2], Birgit Pfitzmann[3]

Abstract. With a fail-stop signature scheme, the supposed signer of a forged signature can prove to everybody else that it was a forgery. Thus the signer is secure even against computationally unrestricted forgers. Until recently, efficient constructions were only known for restricted cases, but at Eurocrypt '92, van Heijst and Pedersen presented an efficient general scheme, where the unforgeability is based on the discrete logarithm.

We present a similar scheme based on factoring: Signing a message block requires approximately one modular exponentiation, and testing it requires a little more than two exponentiations. It is useful to have such alternative constructions in case one of the unproven assumptions is broken.

With all fail-stop signatures so far, the size of the secret key is linear in the number of messages to be signed. In one sense, we prove that this cannot be avoided: The signer needs so many secretly chosen random bits. However, this does not imply that these bits ever have to be secretly stored at the same time: We present a practical construction with only logarithmic secret storage and a less practical one where the amount of secret storage is constant.

We also prove rather small lower bounds for the length of public keys and signatures. All three lower bounds are within a small factor of what can be achieved with one of the known schemes.

Finally, we prove that with unconditionally secure signatures, like those presented by Chaum and Roijakkers at Crypto '90, the length of a signature is at least linear in the number of participants who can test it. This shows that such schemes cannot be as efficient as fail-stop signatures.

1 Introduction and Overview over the Results

Ordinary and Fail-Stop Signatures

Ordinary digital signatures, as introduced in [DH76] and formally defined in [GMR88], allow a person who knows a secret key to make signatures that everybody else can verify with a corresponding public key. Such signatures can only be computationally secure: A forger with unrestricted computing power can always forge signatures of other persons. The security of the schemes relies on the fact that a realistic forger has not enough time to carry out brute-force search and the assumption that there is no really efficient algorithm to compute forgeries.

1 CWI, Kruislaan 413, NL-1098 SJ Amsterdam

2 Aarhus Universitet, Computer Science Department, Ny Munkegade, DK-8000 Aarhus C, tppedersen@daimi.aau.dk

3 Institut für Informatik, Universität Hildesheim, Samelsonplatz 1, W-3200 Hildesheim, pfitzb@informatik.uni-hildesheim.de

With fail-stop signatures, introduced in [WP90] and formally defined in [PW90], unforgeability also relies on a computational assumption. If nevertheless a signature is forged, the alleged signer can prove that the signature is a forgery. More precisely, she can prove that the underlying computational assumption has been broken. This proof may fail with an exponentially small probability, but the ability to prove forgeries does not rely on any cryptographic assumption and is independent of the computing power of the forger. Thus a polynomially bounded signer can be protected from an all-powerful forger. Moreover, after the first forgery, all participants, or the system operator, know that the signature scheme has been broken, so that it can be stopped. This is where the name "fail-stop" comes from.

For more details about possible benefits of fail-stop signatures in applications, e.g., in electronic payment systems, and possible advantages for the acceptability of digital signatures in law, see [PW91, P91].

Previous Constructions

So far, there have been three significantly different results about fail-stop signatures.

Theoretically, fail-stop signature schemes are known to exist if claw-free pairs of permutations (not necessarily with trap-door) exist; see [BPW91, PW91] for descriptions and [PW90] for a proof. In particular, this shows that fail-stop signatures exist if factoring large integers or computing discrete logarithms is hard. The construction uses one-time signatures, similar to [L79], i.e., messages are basically signed bit by bit. Therefore, although messages can be hashed before signing and tree-authentication is used (similar to [M80]), this general construction is not very efficient.

There is an efficient variant especially suited for making clients unconditionally secure in on-line payment systems, see [P91]. However, in this scheme, all signatures by one client (with one key) must have the same recipient, like the bank in a payment system. Furthermore, signing is a 3-round protocol between the signer and the recipient.

The first efficient general fail-stop signature scheme was presented in [HP92]. The unforgeability relies on a discrete logarithm assumption. Signatures for one message block are about as efficient as with RSA. Messages can be hashed before signing. In contrast to RSA, the signer needs some new random bits for each new signature, and tree authentication is needed to keep the public keys short. However, fast hash functions can be used without reducing the security of the signer.

Related Types of Systems

In [CHP92], unconditional security for the signer was achieved in undeniable signatures (cf. [CA90]). The construction was the first not to use bit-by-bit signing. Apart from the usual differences between ordinary and undeniable signatures, this scheme differs from efficient fail-stop signatures in two ways: First, although the signatures themselves are efficient, the verification protocol requires quite a lot of computation, because it needs σ challenges (similar to signatures) to achieve an error probability of $2^{-\sigma}$. Secondly, if the computational assumption is broken, signers can disavow signatures, but there is no way for the recipient to prove to a third party that this is due to cheating (whereas with fail-stop signatures, third parties can distinguish whether the signatures just don't pass the test, or whether they are disavowed due to a proof of forgery). In particular, one cannot stop the scheme as soon as this happens.

In [CR91], unconditionally secure signatures were introduced, i.e., signature-like schemes where both the signer and the recipient are unconditionally secure. In [PW92], a transferable version was presented, i.e., signatures can be passed on from one recipient to another, and security against active attacks on recipients was achieved; such attacks must be considered because the recipients, too, have secret information in such schemes. With these extensions, unconditionally secure signatures could in principle replace other signatures in many applications. So far, however, they are too inefficient to be used in practice: They require a complicated interactive key generation protocol in many rounds, and signatures are very long. Hence they cannot replace ordinary or fail-stop signatures at present.

Overview over the New Results

We present two new constructions of efficient fail-stop signatures (Ch. 3 and 5) and some general lower bounds (Ch. 4).

The first construction has similar properties to that from [HP92], but the unforgeability is based on factoring instead of the discrete logarithm. Signing a message block requires about one modular exponentiation, testing a little more than two. Key exchange is in general more complicated than for the discrete logarithm scheme. Nevertheless, with all types of cryptographic systems it is useful to have alternative constructions, in case one of the unproven assumptions is broken.

The second construction and the first lower bound deal with the fact that in all fail-stop signature schemes so far, the size of the secret key is linear in the number of messages to be signed. We show that in the sense of secret storage needed, this can be avoided, whereas in the sense of choosing secret random bits, it cannot.

Constructions with small secret storage may be important since secret storage is quite hard to realize: One needs a more or less tamper-proof device. In contrast, information can quite easily be stored just securely, since one can distribute several copies. (Note that even ordinary digital signatures assume that a lot of information can be stored securely, since all signatures must usually be stored by their recipients.) In Ch. 5, we present an efficient construction where the size of the secret storage space is logarithmic in the number of messages to be signed, and an otherwise less efficient variant where this size is constant.

For the lower bounds, we assume that the probability that a forgery cannot be proved is smaller than $2^{-\sigma}$ for some security parameter σ, and that the recipient wants a similar level of security at least against simple brute-force forging algorithms. Then the most important result we obtain about fail-stop signatures is:

- If N messages are to be signed, the signer needs at least $(N + 1)(\sigma - 1)$ secretly chosen random bits. More precisely, this is a lower bound for the entropy of her secrets, given the public key.

Additionally, we show two more lower bounds for fail-stop signatures. They are not much larger than similar bounds for ordinary digital signatures would be, since they concern parameters where the difference between current fail-stop signatures and ordinary digital signatures is already quite small.

- The entropy, and hence the length, of a signature is at least $2\sigma - 1$, and the entropy of the public key is at least σ, even if a prekey is already given, i.e., some information trusted by recipients and chosen before the signer chooses her actual keys.

Finally, we show that unconditionally secure signatures cannot be as efficient as fail-stop signatures:

- The entropy (and thus the length) of each unconditionally secure signature that can be tested by M participants, including those that only have to settle disputes, is at least $M \bullet \sigma$.

2 Brief Sketch of Definitions

Like an ordinary digital signature scheme, a fail-stop signature scheme contains a method to generate secret and public keys and algorithms *sign* for signing messages and *test* for testing signatures. Additionally, there is an algorithm *prove*, which the signer uses to produce a proof of forgery from a forged signature, and an algorithm *proof_test*, which everybody else uses to test if something really is a proof of forgery.

A secure fail-stop signature scheme has the following properties, where 2. is a consequence of the others:

1. If the signer signs a message correctly, then the recipient accepts the signature.
2. A polynomially bounded forger cannot make signatures that pass the signature test.
3. If an unrestricted forger succeeds in constructing a signature that passes the signature test, then with “overwhelming” probability, the signer can produce a proof of forgery that convinces any third party that a forgery has in fact occurred (i.e., the output of *prove* passes *proof_test*).
4. A polynomially bounded signer cannot make a (false) signature that she can later prove to be a forgery.

The basic idea to achieve these properties is that (exponentially) many secret keys correspond to each public key, and different secret keys give different signatures on the same message. The signer knows exactly one of these secret keys and can only construct one of the possible signatures on a given message. However, even an arbitrarily powerful forger does not have sufficient (Shannon) information to determine which of the many possible signatures the signer can construct on a new message. Consequently, with very high probability a forged signature will be different from the signature that the signer would have constructed. The knowledge of two different signatures on the same message then yields a proof of forgery.

Since there must be security for both signers (see 3.) and recipients (see 4.), both take part in key generation. Usually, the recipient (or all possible recipients together, or a device trusted by all recipients) chooses a value called prekey, such as a number that the signer cannot factor, and then the signer chooses the real secret and public key based on this prekey. However, we prove the lower bounds for an arbitrary key generation protocol.

There are also two security parameters: σ determines that the probability of unprovable forgeries is smaller than $2^{-\sigma}$, and k is the parameter for the cryptographic security of the recipient. Usual choices of σ may be between 40 and 100, whereas k, if it is the binary length of numbers that should be hard to factor in Ch. 3, must be larger than 500.

Remark: Note that it is not a matter of the definition how one acts if a proof of forgery occurs. In particular, instead of making signers unconditionally secure by invalidating signatures after proofs of forgery, one could leave the responsibility with the signer. Then one has all the properties of an ordinary digital signature scheme, plus the possibility to stop after forgeries. (This shows that fail-stop signatures are a strictly stronger notion.)

Furthermore, the current definition does not specify for how much of a system a particular proof of forgery is valid. As long as forging even one signature is provably as hard as, say, factoring, one should stop the whole scheme after any forgery, because if one signature has been forged, the same forger can probably forge them all. Therefore, the constructions usually assume that there is just one type of proof of forgery. However, it is no problem to make proofs of forgery specific to the keys of individual signers or even (although currently with some loss in efficiency) to each particular signature. ♦

For a complete formal definition, see [PW90]. In this abstract, we will only make those parts more precise that are actually needed in the proofs of the lower bounds.

3 Efficient Fail-Stop Signatures based on Factoring

This section presents a fail-stop signature scheme based on the assumption that it is infeasible to factor large integers. To emphasize the generality of the construction, the scheme is first described in general terms. Like in [HP92], we first present a version for signing just one message block.

3.1 General Structure of the Construction

The following construction generalizes that from [HP92]. We base it on so-called **bundling homomorphisms**, i.e., functions h with the following properties:

1. h is a homomorphism between two Abelian groups.
2. Given an image $h(a)$, there exist at least 2^τ possible preimages.
3. It is infeasible to find collisions, i.e., two different values that are mapped to the same value by h.

More precisely, there must be a family of such functions and groups, and a key generation algorithm that selects a particular function h, given τ and a security parameter k. One also needs efficient algorithms for the group operations and to choose random elements.

Now we define all the components of a fail-stop signature scheme (cf. Ch. 2):

- Prekey: The recipient selects a function h from the family. Let the domain be G and the range H.
- Prekey test: The recipient must prove that his choice of h was correct, or at least that his h is in a set of functions with Properties 1 and 2 (which are needed for the security of the signer).
- Secret key: $sk := (sk_1, sk_2)$, where sk_1 and sk_2 are chosen at random from G.
- Public key: $pk := (pk_1, pk_2)$, where $pk_i = h(sk_i)$ for $i = 1, 2$.

- Signing: $sign(sk, m) = sk_1 \bullet sk_2^m$ for messages m from a subset (to be defined) of $\mathbb{Z}$.
- Test: $test(pk, m, s) = ok \;:\Leftrightarrow\; pk_1 \bullet pk_2^m = h(s)$.
- Proving forgeries: Given a forged signature sf on a message m^*, the signer computes $s = sign(sk, m^*)$, and if $s \neq sf$, she uses the pair (s, sf) as a proof of forgery.
- Testing proofs of forgery: Given two elements of G, verify that they collide under h.

Theorem 1: Independently of the choice of h, the following holds:

1. Correct signatures pass the test: $h(s) = h(sk_1 \bullet sk_2^m) = pk_1 \bullet pk_2^m$.
2. A polynomially bounded signer cannot construct a signature and a proof that it is a forgery.
3. If sf is a forged signature on m^* and $sf \neq sign(sk, m^*)$, then the signer obtains a valid proof of forgery. ♦

Proof: Follows easily from the definitions. □

This theorem shows that the general scheme is secure for the recipients, and that it is also secure for the signer if even an all-powerful forger cannot guess a correct signature $s = sign(sk, m^*)$, except with a very small probability. In order to estimate the probability with which a forger can guess s, first note that the public key contains no information about which of at least $2^{2\tau}$ possible secret keys the signer actually has. However, after having received a signature on a message m, the forger has more information about sk. Theorem 2 gives a condition for when this information is not sufficient to construct new signatures that the signer cannot repudiate:

Theorem 2: Let pk, a signature $s = sign(sk, m)$, and a message $m^* \neq m$ be given, and let $m' := m^* - m$. Whatever value sf an all-powerful forger selects as a forged signature on m^*, the probability that it is correct is at most $|T| / 2^\tau$, where

$$T := \{d \in G \mid h(d) = 1 \wedge d^{m'} = 1\} = \{d \mid h(d) = 1 \wedge \mathrm{ord}(d) \mid m'\}.$$

(The probability is given by the secret keys that are still possible when pk and s are known.) ♦

Proof: The set of possible secret keys is

$$SK^* := \{(sk_1, sk_2) \in G \times G \mid h(sk_1) = pk_1 \;\wedge\; h(sk_2) = pk_2 \;\wedge\; sk_1 \bullet sk_2^m = s\}$$
$$= \{(s / sk_2^m, sk_2) \mid h(sk_2) = pk_2\},$$

because of the homomorphism property; and the size of this set is 2^τ. The attacker is successful if

$$sk_1 \bullet sk_2^{m^*} = sf.$$

For keys from SK^*, this equation is equivalent to $sk_2^{m'} = sf / s$. This equation may be unsolvable, but if there is any solution sk_2^*, then the set of all solutions in SK^* is

$$\{(s / sk_2^m, sk_2) \mid h(sk_2 / sk_2^*) = 1 \;\wedge\; (sk_2 / sk_2^*)^{m'} = 1\}.$$

Hence the number of solutions is $|T|$, and the attacker is successful with the claimed probability. □

Consequently, in order to estimate the probability of successful forgeries we must find the size of T. This size depends on the chosen family of homomorphisms.

3.2 The Special Case with Factoring

Our family of bundling homomorphisms was defined in [BPW91], using ideas from [GMR88, G87]: A member of the family is characterized by τ and a k-bit integer $n = pq$, with p, q prime and $p \equiv 3$ and $q \equiv 7 \bmod 8$. We omit τ and n in the following. The groups are

$$H = \pm QR/\{\pm 1\}, \text{ and } G = \mathbb{Z}_{2^\tau} \times H,$$

where QR denotes the group of quadratic residues modulo n, and the operation on G is given by

$$(a, x) \circ (b, y) := ((a + b) \bmod 2^\tau, x \bullet y \bullet 4^{(a + b) \operatorname{div} 2^\tau}).$$

Elements of H are represented by numbers between 0 and $n/2$; H is used instead of QR because membership can be tested efficiently. The unit element of G is (0,1). The homomorphism is given by

$$h((a, x)) = \pm (4^a \bullet x^{2^\tau}).$$

Theorem 3: The construction described above is a family of bundling homomorphisms. Properties 1 and 2 even hold for any odd n. Furthermore, if n is chosen correctly or at least as $n = p^r q^s$ where p and q are correct and r, s odd, then for any a, z, there exists exactly one x so that $h((a, x)) = z$. ♦

Proof: See [BPW91]. The last sentence is only proved for correctly chosen n there, but the same proof is valid for the more general form. □

To use these homomorphism in a fail-stop signature scheme according to Sect. 3.1, let the message space be $\{0, \ldots, 2^\rho - 1\}$ for some ρ and $\tau := \rho + \sigma$. As an efficient prekey test, we use the protocol from [GP88] and a test that $n \equiv 5 \bmod 8$. Actually, this does not completely prove that n is of the correct form, but it ensures that $n = p^r q^s$ where p and q are correct and r, s odd.

Theorem 4: With the definitions made above, the probability of undetected forgery is at most $2^{-\sigma}$. ♦

Proof: According to Th. 1 and 2, it only remains to prove $|T| \le 2^\rho$. Note that in G

$$(a, x)^{m'} = (0, 1) \Rightarrow m' \bullet a \bmod 2^\tau = 0 \Rightarrow \operatorname{ord}(a) \mid m'.$$

Hence $$T \subseteq \{(a, x) \in G \mid h((a, x)) = 1 \wedge \operatorname{ord}(a) \mid m'\}.$$

According to Th. 3, for each a, there is exactly one x such that $h((a, x)) = 1$. Thus

$$|T| \le |\{a \in \mathbb{Z}_{2^\tau} \mid \operatorname{ord}(a) \mid m'\}| = \gcd(2^\tau, m'). \quad (3)$$

By the choice of message space, every pair of messages $m \neq m^*$ satisfies $|m - m^*| < 2^\rho$ and therefore $\gcd(2^\tau, m - m^*) < 2^\rho$. □

As to efficiency, first note that a multiplication in G is mainly one modular multiplication, since the exponent of 4 is 0 or 1, and a multiplication by 4 can be replaced by shifts and subtractions. We can choose any fixed message length ρ; long messages are hashed before signing. Since even the hash functions as secure as factoring from [D88] take only one multiplication per message bit, i.e., not more than signing or testing, one should always hash messages as short as possible. Thus ρ is determined by the size of the output of the hash function. In the following table of the efficiency of signing one message block, we assume $\rho = k$. If one trusts a faster hash function, or in applications where only short messages are signed, one can still gain efficiency by making ρ smaller.

sign:	k multiplications
test:	$2k + \sigma$ multiplications
Length of *pk*:	$2k$
Length of *sk*:	$4k + 2\sigma$
Signature length:	$2k + \sigma$

To sign several messages, one can use tree authentication as in [PW91, HP92], after [M80]. Note that key exchange is more efficient in [HP92] because the choice of the prekey is just a choice of random numbers, and no prekey test is necessary even if there is no trusted device to choose the prekey.

4 Lower Bounds

The idea of each of our proofs will first be described informally. For the formal sketches, we assume the reader knows the notions of conditional entropy, $H(X \mid Y)$, and mutual information, $I(X; Y)$; see [S49, G68 Sect. 2.2, 2.3]. Like in [G68], we use capital letters for random variables and small letters for corresponding values, and abbreviate $P(X = x)$ by $P(x)$ etc. The formula we need most is the **chain rule** to add entropies:

$$H(Y, Z \mid X) = H(Y \mid X) + H(Z \mid Y, X).$$

Additionally, when we know that the probability that something can be guessed correctly is small, and want to derive that a conditional entropy is large, we often need **Jensen's inequality** for the special case of the logarithm [F71]: If $p_i \geq 0$ and $x_i > 0$ for all i, and the sum of the p_i's is 1, then

$$\log\left(\sum_i p_i x_i\right) \geq \sum_i p_i \log(x_i).$$

4.1 Secret Keys, or Rather, Secret Random Choices

The basic reason why the signer needs a lot of secretly chosen random bits is:

1. Even an arbitrarily powerful forger must not be able to guess the signer's correct signatures.
2. Since this holds for each additional signature, even when some signatures are already known, the entropy of each new signature must be large, and therefore the overall entropy of the signer's secrets is large.

However, sometimes the forger *does* know correct signatures on new messages. For instance, in schemes with message hashing, the forger knows the signatures on all messages with the same hash value as the original message. (Then the collision counts as a proof of forgery.) Hence Statement 1 does not hold absolutely. Instead, we will derive an average version as follows:

1.1 With high probability, the signer should not obtain proofs of forgery if she applies *prove* to her correct signatures; otherwise she could cheat the recipient. (The probability is over the choice of the keys; we will see that we can leave the messages fixed.)

1.2 Thus, on average, even an all-powerful forger must not be able to guess those correct signatures.

In 1.1, the recipient's security is needed. (Note that the desired theorem cannot possibly be proved from the signer's security alone. As a counterexample consider that the signer were allowed to disavow all signatures in an ordinary digital signature scheme; then she would be unconditionally secure without many random bits, but the recipient would not be secure at all.) This is a problem, since the recipient's security, like all computational cryptographic definitions, is only defined asymptotically. It says: For any polynomial-time algorithm $\tilde{A}$ and any c, there exists k_0 so that the probability that $\tilde{A}$ successfully cheats the recipient is smaller than k^{-c} for all $k \geq k_0$. Thus, in a certain sense, we can only derive lower bounds for $k \geq k_0$, for an unknown k_0. This may seem unsatisfactory: Nobody would have doubted that we need arbitrarily long keys if we make k sufficiently large.

However, note that the real purpose of our lower bounds is to say "whenever we have certain requirements on the security, then we have to pay the following price in terms of efficiency". In this section, this is more precisely: "If the signers want the probability of unprovable forgery to be at most $2^{-\sigma}$, and the recipients want some security, too, then at least the following number of random bits is needed (as a function of σ and the security of the recipients)".

To quantify the security of the recipient, it suffices for our purpose to consider the case of Statement 1.1 above, i.e., we consider the probability with which the signer can prove that her own correct signatures are forgeries just by applying the algorithm *prove* to them. In practice, one has to require this probability to be at most, say, 2^{-20}, or, more generally, $2^{-\sigma^*}$ for some σ^*. We will prove the lower bounds as a function of this parameter σ^* (in addition to the σ from the signer's security). To formulate the theorem precisely, we need some more notation and partial definitions:

- **Key exchange and probability space:** Key exchange is a protocol G with inputs σ, k, and the number of messages to be signed, N, all in unary. The output is a pair (sk, pk) of a secret and a public key. For the lower bounds, we only need the case where all parties execute G honestly, and we always consider a fixed triple of parameters. Then the probabilities of sk and pk are uniquely determined, and we can define corresponding random variables SK and PK.

  Without loss of generality, we assume that all random bits that the signer needs are already in sk, so that *sign* is deterministic, and so are *test*, *prove*, and *proof_test*. Thus, the underlying probability space for all probabilities is that of the secret random bits used in the key exchange.

- **Signing:** We make the lower bounds quite general by permitting the signer to use memory in a general way, i.e., signatures may depend on all previously signed messages. We even allow testing to be equally general, although this is only useful when there is a single recipient.

- **Probability that the signer can disavow her correct signatures:** For every message sequence $\underline{m} = (m_1,\ldots, m_{N+1})$, we define a polynomial-time algorithm $\tilde{A}_{\underline{m}}$ to describe what a dishonest signer would do to disavow her own signatures: After executing G correctly, i.e., on input sk, she first signs $m_1, \ldots, m_N$ correctly. Then, since m_{N+1} is one message too much, she signs it as if she had not signed m_N. From each of these signatures, together with sk and the history of preceding signatures, she tries to compute a proof of forgery using *prove*. (This algorithm should be rather useless!)

If the fail-stop signature scheme, N, and σ are fixed, we say that k is large enough to provide the security level σ^* for the recipient against $\tilde{A}_{\underline{m}}$ if the success probability of $\tilde{A}_{\underline{m}}$ is at most $2^{-\sigma^*}$.[4]

The formal version of the theorem is therefore:

Theorem 5: Let a fail-stop signature scheme with actual parameters σ and N and a security level σ^* be given. Let $\sigma' := \min(\sigma, \sigma^*)$. Then for all k sufficiently large to provide the security level σ^* for the recipient against an algorithm $\tilde{A}_{\underline{m}}$ for any sequence $\underline{m}$ of $N+1$ pairwise distinct messages[5],

$$H(SK \mid PK) \;\geq\; (N+1)(\sigma'-1). \qquad \blacklozenge$$

Since $\underline{m}$ is fixed within the theorem, we can omit it in the proof. Let S_i denote the random variable of the signature on the i-th message of $\underline{m}$, and $Hist_i$ that of the history of the first i signatures. The following lemma formalizes that on average, correct signatures cannot be guessed:

Lemma 1: With the same notation as in Th. 5, for each $i \leq N+1$:

$$H(S_i \mid PK, Hist_{i-1}) \;\geq\; \sigma' - 1. \qquad \blacklozenge$$

We must omit the proof of Lemma 1 in this abstract. However, it proceeds along the informal description, exploiting the difference that correct signatures can usually not be disavowed, whereas guessed ones can, with an application of Jensen's inequality at the end.

Proof of Th. 5: First we use Lemma 1 to show by induction over i that the entropy of all signatures together is large. Remember $Hist_i = (S_1, \ldots, S_i)$. Hence, we show for all $i \leq N+1$:

$$H(Hist_i \mid PK) \;\geq\; i \bullet (\sigma' - 1). \qquad (1)$$

For $i = 1$, (1) is just Lemma 1. And if (1) has already been proved for $i-1$, then it holds for i because

$$\begin{aligned} H(Hist_i \mid PK) &= H(S_i \mid PK, Hist_{i-1}) + H(Hist_{i-1} \mid PK) \\ &\geq (\sigma'-1) + (i-1)(\sigma'-1) = i \bullet (\sigma'-1). \end{aligned}$$

We now use that signing is deterministic, i.e., SK uniquely determines $Hist_{N+1}$. This implies $H(Hist_{N+1} \mid PK, SK) = 0$, and therefore with the chain rule

$$\begin{aligned} H(SK \mid PK) &= H(SK, Hist_{N+1} \mid PK) - H(Hist_{N+1} \mid PK, SK) = H(SK, Hist_{N+1} \mid PK) \\ &\geq H(Hist_{N+1} \mid PK) \;\geq\; (N+1)(\sigma'-1). \qquad \square \end{aligned}$$

4.2 Signatures and Public Keys

Signatures and public keys are not much longer in current fail-stop signature schemes than in ordinary signature schemes. Hence the lower bounds are very small, too.

The basic idea about the length of a signature is:

---

4 The formal definition of the recipient's security immediately implies the existence of k_0 such that all $k \geq k_0$ have this property. We have now bypassed the problem that we do not know how large k_0 is because we just know that it must be large enough in a practical application.

5 Note that we only require security against $\tilde{A}_{\underline{m}}$ for one message sequence $\underline{m}$. The contrary is that *all* these algorithms work.

a) First, there must be at least 2^{σ} acceptable signatures; otherwise the correct signature could be guessed too easily.

b) Secondly, it must be hard for a forger to guess signatures at all. Thus the density of the set of acceptable signatures within the signature space should be small, e.g., at most $2^{-\sigma^*}$.

Hence we expect the size of the signature space to be at least $2^{\sigma+\sigma^*}$. Indeed, we prove more generally that the entropy of each signature is at least $\sigma+\sigma^*$. What has to be done is:

- Since the forger in (b) is computationally restricted, we must show that he could guess acceptable signatures *efficiently* if their density was too high.
- As in 4.1, we must require that k is sufficiently large so that a concrete version of the asymptotic security against forgery holds.
- We must express the idea with the density in information-theoretic terms.

For this, we first define a simple algorithm $\tilde{F}_m$ that tries to guess signatures on a message m (in a rather stupid way): $\tilde{F}_m$ just chooses its own key pair (sk^*, pk^*) and signs m with sk^*.

Theorem 6: Assume a fail-stop signature scheme with actual parameters k, σ, N provides the security level σ^* against forgery by an algorithm $\tilde{F}_m$.

1. Let S be the random variable of the signature. Then
$$\mathrm{H}(S) \;\geq\; \sigma' + \sigma^* - 1.$$
(If the scheme is not memory-less, we obtain the same result for later messages by using the last message of a message sequence $\underline{m}$.)

2. $$\mathrm{H}(PK) \geq \sigma^*. \qquad \blacklozenge$$

The following lemma formalizes the density argument. The fact that the number of possible signatures, given the public key, is much smaller than the complete signature space is generalized as follows: The public key contains a lot of information about the correct signature.

Lemma 2: With the same notation as in Th. 6,
$$\mathrm{I}(S;\, PK) \geq \sigma^*. \qquad \blacklozenge$$

The proof must be omitted in this abstract.

Proof of Th. 6: Lemma 2 means $\mathrm{H}(S) - \mathrm{H}(S \mid PK) \geq \sigma^*$; and a special case of Lemma 1 is $\mathrm{H}(S \mid PK) \geq \sigma' - 1$. Consequently, $\mathrm{H}(S) \;\geq\; \mathrm{H}(S \mid PK) + \sigma \;\geq\; \sigma' + \sigma^* - 1$. Furthermore, $\mathrm{I}(S;\, PK) \;\leq\; \mathrm{H}(PK)$. □

For the case with a prekey (cf. Ch. 2), we obtain the same results with an additional condition over K, i.e., $\mathrm{H}(S \mid K) \geq \sigma' + \sigma^* - 1$ and $\mathrm{H}(PK \mid K) \geq \sigma^*$. If, as usual in such schemes, PK is a function of SK, we obtain one more result by applying the chain rule to the last formula and Th. 5:

Theorem 5*: In a fail-stop signature scheme with prekey, and where the public key is a function of the secret key, and with the same notation as in Th. 5 and 6,
$$\mathrm{H}(SK \mid K) \;\geq\; (N+1)(\sigma'-1) + \sigma^* \;\geq\; (N+2)(\sigma'-1). \qquad \blacklozenge$$

4.3 Unconditionally Secure Signatures

Unconditionally secure signature schemes could be achieved by replacing the globally known public key *pk* (which implied that an all-powerful forger could find acceptable signatures by brute-force search) by different test keys t_x for each recipient x. So far, this has made key exchange complicated and signatures long.

Essentially, we prove that such signatures must indeed be at least as long as if they consisted of an independent part for each test key, i.e., they cannot be shortened by a suitable combination. Assume M people may want to test a signature (as a recipient, or to settle a dispute), and that the probability for successful forgeries is to be $\leq 2^{-\sigma}$. The basic idea is: If some participants want to forge a signature on m, they can determine the set of signatures acceptable under all their test keys. Still, within this set, the density of signatures that another participant accepts must not exceed $2^{-\sigma}$. Inductively, this implies that the size of the original signature space must be at least $2^{M\sigma}$.

In Theorem 7, we generalize this to entropies, and we show that it holds for every signature, even if signing is not memory-less.

Theorem 7: Consider an unconditionally secure signature scheme with M recipients where N messages can be signed and the probability of successful forgery is $\leq 2^{-\sigma}$. For any given message sequence $\underline{m}$, let S_i denote the random variable of the signature on the i-th message of $\underline{m}$, and $Hist_i$ that of the history of the first i signatures. Then

$$\mathrm{H}(S_i \mid Hist_{i-1}) \;\geq\; M\sigma. \qquad \blacklozenge$$

The basic idea for the proof is formalized similar to Lemma 2: Even when some test keys are known, any other test key still gives a lot of information about the correct signature.

Lemma 3: With the same notation as in Th. 7: For any set X of participants and $y \notin X$, if T_X denotes the joint random variable of the test keys of X:

$$\mathrm{I}(S_i; T_y \mid T_X, Hist_{i-1}) \geq \sigma. \qquad \blacklozenge$$

Again, we must omit the proof in this abstract.

Proof of Th. 7: Lemma 3 means $\mathrm{H}(S_i \mid T_X, Hist_{i-1}) \;\geq\; \sigma + \mathrm{H}(S_i \mid T_{X\cup\{y\}}, Hist_{i-1})$. With induction over the size of X, one easily obtains the desired result. □

From Th. 7 and Lemma 3, with induction over i similar to that in Th. 5, we can also obtain

$$\mathrm{H}(SK) \geq (N+1)M\sigma \;\wedge\; \mathrm{H}(T_y \mid T_X) \geq (N+1)\sigma.$$

5 Fail-Stop Signatures with Small Secret Storage

To show that the signer needs far less secret storage than the number of secret bits she must choose according to Th. 5, we proceed in two steps: First we show a simple construction where only a small amount of secret storage is needed at the start, i.e., directly after key exchange. Then we add additional measures so that the amount of secret storage is small all the time.

The basis of this section is a fail-stop signature scheme for signing just one message of arbitrary length. We use the scheme described in Section 3.1 combined with message hashing. Hence the construction works for the schemes from [HP92] and Section 3.2.

(a) Small amount of secret information at the start: Use "top-down" tree-authentication similar to [M88, GMR88]. (Note that a different "bottom-up" version, which is a little more efficient if one does not consider secret storage space, was normally used with fail-stop signatures so far.) Let a prekey, i.e., a bundling homomorphism h, be given. The signer starts with one pair of a secret and a public key at the root of the tree. Then she creates two children, each with a new key pair, and uses the old secret key to sign a message containing the two new public keys. For each of the two new nodes, she again constructs two children in the same way, and so on. Messages are signed using the secret keys at the leaves of the tree, and a complete signature is one branch of these original signatures.

During key exchange, only the root of the tree has to exist, and to sign the first message, only the keys on the left-most branch and their immediate other children have to be generated. Figure 1 shows the situation after the first message, $m_{0\ldots0}$, has been signed.

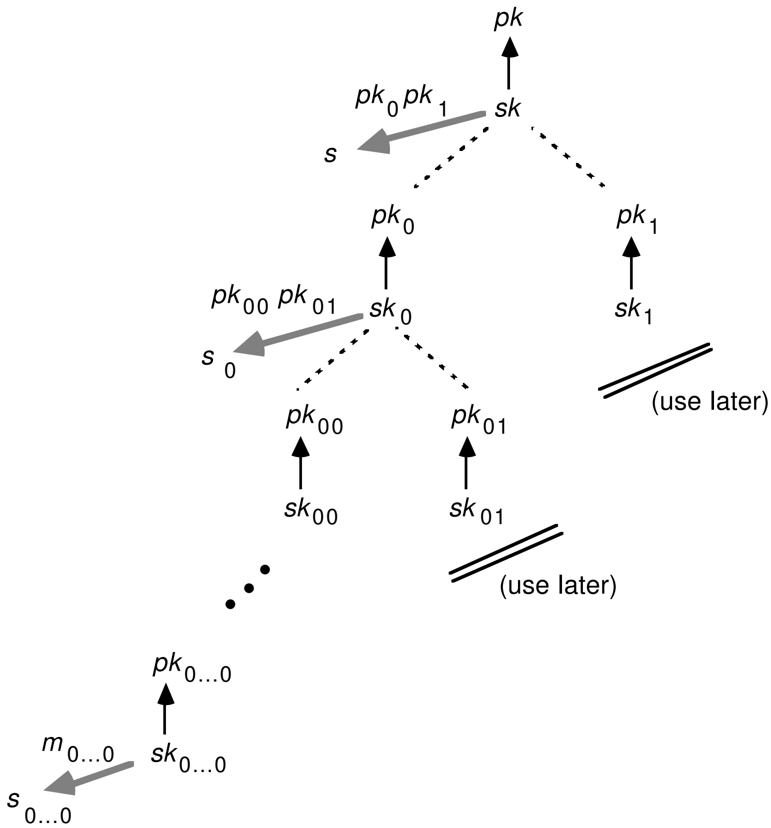

Figure 1 Fail-stop signature scheme with "top-down" tree authentication.
Thin black arrows denote the computation of a public key from a secret key in a basic scheme to sign just one message (like in Ch. 3, together with message hashing), broad grey arrows denote signatures in the basic scheme, and dotted lines just indicate a tree, but are not related to a computation.

At any time, just one branch of the tree has to be stored for signing. However, so far, the individual secret keys sk_j that are used up, i.e., that are no longer needed for signing, must be stored until the end so that forgeries at any node can be proved.

(b) Small amount of secret storage altogether: The basic idea to reduce secret storage further is to store values sk_j that are used up in encrypted form and to store just the key secretly. However, information-theoretically secure encryption is needed, and a one-time pad is of no use because the key would be just as long as the encrypted message. Hence special care must be taken that each individual sk_j is still secret enough, although information about the ensemble of sk_j's may become known.

If the individual sk_j's are formed according to Section 3.1, this is achieved by the following additional steps:

1. Initially, the signer chooses a value $e \in G$ randomly as an encryption key. She keeps e secret all the time.
2. Whenever the signer has used up a value $sk_j = (sk_{j,1}, sk_{j,2})$ by signing a message m_j, she proceeds as follows:
   - She encrypts $sk_{j,2}$ as $c_j := sk_{j,2} \bullet e$.
   - She stores m_j, the signature s_j, and the ciphertext c_j securely, but not necessarily secretly.

Theorem 8: If the tree construction described in (a) is applied to a secure fail-stop signature scheme constructed according to Section 3.1 together with message hashing, and the additional steps described in (b) are taken, then we have a secure fail-stop signature scheme again. ♦

Proof: First, the signer can reconstruct any secret key sk_j if she needs it to prove a forgery: She decrypts $sk_{j,2} = c_j / e$ and then recomputes $sk_{j,1} = s_j / sk_{j,2}{}^m$, where m is the hash value of m_j.

Hence, whenever a signature for a node j is forged and it is different from the signature the signer would have produced for the same message, the signer can prove this forgery just as in Section 3.1. Furthermore, every complete forgery sf (i.e., a branch of the tree) must be linked into the correct tree somewhere, i.e., it contains at least one such forgery at a node j for the correct pk_j.

Thus it remains to show that the additional information stored securely does not help a forger to find exactly the signature that the signer would have produced at node j. This signature depends only on sk_j (i.e., not on the values sk_l at other nodes). In the original scheme, the set of possible values sk_j from the point of view of a forger was $SK_j{}^* = \{(s_j / sk_{j,2}{}^m, sk_{j,2}) \mid h(sk_{j,2}) = pk_{j,2}\}$. Hence it suffices to show that all these values are still possible when the forger has seen c_j and all the other ciphertexts c_l.

Let such a value $sk^*_{j,2}$ be given. It corresponds to exactly one key $e^* = c_j / sk^*_{j,2}$. This implies that the other plaintexts must be $sk^*_{l,2} = c_l / e^* = sk^*_{j,2} \bullet c_l / c_j$. The only question is if these are possible plaintexts, i.e., if $h(sk^*_{l,2}) = pk_{l,2}$. On the one hand, $h(sk^*_{l,2}) = h(sk^*_{j,2}) \bullet h(c_l) / h(c_j) = pk_{j,2} \bullet h(c_l) / h(c_j)$. On the other hand, $h(c_l) = h(sk_{l,2}) \bullet h(e) = pk_{l,2} \bullet h(e)$ and $h(c_j) = pk_{j,2} \bullet h(e)$, hence $h(c_l) / h(c_j) = pk_{l,2} / pk_{j,2}$. This yields $h(sk^*_{l,2}) = pk_{l,2}$. □

Consequences: If this construction is applied to a usual complete tree, then it is very practical, and at any time, only e and the secret keys that have been marked "use later", i.e., at most one per level of the tree, must be stored secretly. This is a logarithmic amount.

If we use a list-like tree, i.e., the left child of each node is a real message, we only need two sk_j's at any time. However, later signatures are very long. Thus the list-like version should only be used with a fixed recipient, who can store the part of the list he already received, like in [P91].

One can also use trees of other forms or combine it with other methods to sign several messages from [HP92].

6 Conclusion

We have constructed efficient fail-stop signatures based on the assumption that factoring large integers is hard, giving an alternative to the previous scheme based on a discrete logarithm assumption. We also presented a construction which only needs a small amount of secret storage space, whereas in all previous constructions, a secret key whose length was linear in the number of signatures to be issued was stored all the time.

On the other hand, we proved that there is a definite difference to ordinary digital signatures in that the signer must choose an amount of random bits linear in the number of signatures to be issued. Finally, we showed that there is no hope that unconditionally secure signatures can become as efficient as fail-stop signatures, because the length of each unconditionally secure signature is linear in the number of participants who can test it, whereas the length of a fail-stop signature (or an ordinary digital signature) does not depend on this number.

Acknowledgements

It is a pleasure to thank Joachim Biskup, Gerrit Bleumer, David Chaum, Andreas Pfitzmann, and Michael Waidner for interesting discussions.

References

[BPW91] Gerrit Bleumer, Birgit Pfitzmann, Michael Waidner: A remark on a signature scheme where forgery can be proved; Eurocrypt '90, LNCS 473, Springer-Verlag, Berlin 1991, 441-445.

[CA90] David Chaum, Hans van Antwerpen: Undeniable signatures; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 212-216.

[CHP92] David Chaum, Eugène van Heijst, Birgit Pfitzmann: Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 470-484.

[CR91] David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures; Crypto '90, LNCS 537, Springer-Verlag, Berlin 1991, 206-214.

[D88] Ivan Bjerre Damgård: Collision free hash functions and public key signature schemes; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 203-216.

[DH76] Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644-654.

[F71] William Feller: An Introduction to Probability Theory and Its Applications, Vol. II (2nd. ed.); John Wiley & Sons, New York 1971.

[G68] Robert G. Gallager: Information Theory and Reliable Communication; John Wiley & Sons, New York 1968.

[G87] Oded Goldreich: Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme; Crypto ’86, LNCS 263, Springer-Verlag, Berlin 1987, 104-110.

[GMR88] Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

[GP88] Jeroen van de Graaf, René Peralta: A simple and secure way to show the validity of your public key; Crypto ’87, LNCS 293, Springer-Verlag, Berlin 1988, 128-134.

[HP92] Eugène van Heijst, Torben Pryds Pedersen: How to Make Efficient Fail-stop Signatures; Eurocrypt ’92, Extended Abstracts, 24.-28. 5. 1992, Balatonfüred, Hungary, 337-346.

[L79] Leslie Lamport: Constructing Digital Signatures from a One-Way Function; SRI Intl. CSL-98, Oct. 1979.

[M80] Ralph C. Merkle: Protocols for Public Key Cryptosystems; Proc. 1980 Symposium on Security and Privacy, Oakland 1980, 122-134.

[M88] Ralph C. Merkle: A digital signature based on a conventional encryption function; Crypto ’87, LNCS 293, Springer-Verlag, Berlin 1988, 369-378.

[P91] Birgit Pfitzmann: Fail-stop Signatures; Principles and Applications; Proc. Compsec ’91, 8th world conference on computer security, audit and control, Elsevier, Oxford 1991, 125-134.

[PW90] Birgit Pfitzmann, Michael Waidner: Formal Aspects of Fail-stop Signatures; Fakultät für Informatik, University Karlsruhe, Report 22/90, Dec. 1990.

[PW91] Birgit Pfitzmann, Michael Waidner: Fail-stop Signatures and their Application; Securicom 91, Paris, 19.-22. March 1991, 145-160.

[PW92] Birgit Pfitzmann, Michael Waidner: Unconditional Byzantine Agreement for any Number of Faulty Processors; STACS 92, LNCS 577, Springer-Verlag, Berlin 1992, 339-350.

[S49] Claude E. Shannon: Communication in the Presence of Noise; Proceedings of the Institute of Radio Engineers 37/1 (1949) 10-21.

[WP90] Michael Waidner, Birgit Pfitzmann: The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability; Eurocrypt ’89, LNCS 434, Springer-Verlag, Berlin 1990, 690.

(Full version: Unconditional Sender and Recipient Untraceability in spite of Active Attacks – Some Remarks; Fakultät für Informatik, University Karlsruhe, Report 5/89, March 1989.)