p

UNIVERSITÄT KARLSRUHE

FAKULTÄT FÜR INFORMATIK

P.O.Box 6980, D 7500 Karlsruhe 1, F. R. Germany

Formal Aspects of Fail-stop Signatures

Birgit Pfitzmann, Michael Waidner
Institut für Rechnerentwurf und Fehlertoleranz

Interner Bericht 22/90
December 1990

Abstract

The unforgeability of conventional digital signatures is necessarily based on complexity theoretic assumptions, i.e. even the most secure schemes can be broken by an unexpectedly powerful adversary. Thus we defined fail-stop signatures: They are as unforgeable as the best conventional signatures, but if a signature is forged, the supposed signer can prove the forgery in an unconditional way, with arbitrarily high probability. We construct fail-stop signatures from any claw-free pairs of permutations.

Here, we mainly present the formal construction.

Table of contents

1 Introduction

1.1 Motivation

Digital signatures are an important cryptographic primitive [DiHe_76, RSA_78]. Their distinguishing feature is that a recipient of a signature can convince everybody else of its validity (in contrast to authentication codes [GiMS_74]). Their security has recently received some attention.

In [GoMR_84, GoMR_88], the optimal security for (conventional) digital signatures was defined. By a **conventional signature** scheme we mean one according to their definition; in particular, each signer Alice publishes a public key *PK*, and something is defined to be Alice's signature iff it passes a test *test*(*PK*, •). This implies that forging signatures is in NP.

Additionally, all known schemes are based on generally trusted, but unproved complexity theoretic assumptions. The original proposal [GoMR_84, GoMR_88] is based on the existence of claw-free pairs of trapdoor permutations. This assumption could first be weakened to any trapdoor functions [BeMi_88], and then to any one-way functions [NaYu_89, SaYu_90]. To use signatures in practice, one must pick out a specific function which is hopefully one-way. Probably one would assume the intractability of integer factoring or of some discrete log. (The most efficient scheme is still that based on factoring from [GoMR_88]).

Fail-stop signatures (**FSSigs**) improve the security in a different way: They unconditionally enable signers to prove forgeries (with arbitrarily high probability). Advantages are:

- The scheme can be stopped, or security parameters increased, as soon as forgeries occur.
- The risk of forged signatures can be divided arbitrarily between signers and recipients. In particular, if signatures become invalid if a proof of forgery has been shown, the signers are unconditionally secure.

We also require that, on cryptographical assumptions, forgeries cannot occur at all, and signers cannot falsely claim that a forgery occurred. Thus, the security definition of FSSigs is strictly stronger than that of a conventional signature scheme.

More introductory remarks can be found in [Pfit1_89, PfWa_91, PfWa1_91].

REMARKS: There have been proposals for conventional signatures, saying that signers notice if forgeries occur, or if the secrecy of their keys is lost, and are then allowed to stop the scheme. However, that would mean total insecurity for the recipients of signatures.

FSSigs have additional practical advantages in public applications like payment systems: First, it is doubtful if people can be forced to take part in systems where they must rely on cryptographical assumptions, with security parameters which they usually cannot choose. Using FSSigs and conventional signatures together (see Ch. 5), one can make clients of such systems unconditionally secure. Secondly, with conventional signatures, even if the cryptographical assumption is true, people might claim that their signatures have been forged, which could lead to considerable public uncertainty.

1.2 Related work

Parallel to this work, CHAUM and ROIJAKKERS have developped a scheme called "unconditionally secure signatures" [ChRo_90]: This is a beautiful idea, but its differences to conventional signatures are quite large: The principle difficulty that forging signatures must be in NP, if there is a known test function for signatures, has been overcome by letting each participant have a different test function which the others do not know. Additionally, Bob's test depends on via how many other recipients the signature has been passed to him; and active attacks on recipients (to find out their test function) are possible. Therefore, in contrast to fail-stop signatures, unconditional signatures cannot just be used in any protocol using signatures, but need special protocols where everybody knows their numbers of predecessors, and where active attacks on recipients are restricted.

In particular, the main scheme of [ChRo_90] has properties similar to the 2-party case here (see Ch. 2). It is not yet known how large the signatures must originally be to be passed along n recipients. (The tricks from [ChRo_90, Chapter 5] can only be used if the signer is good.)

In many cases, FSSigs will also be more efficient than unconditional signatures. Their precomputation phase is simpler, too.

Large parts of this report are based on [Pfit1_89], some on [WaPf_89]. An efficiency improvement (see Ch. 4) has been published in [BlPW_90].

1.3 Overview

In Chapter 2, we discuss the security of FSSigs in general. We also show that FSSigs are secure in the sense of conventional signature schemes.

In Chapter 3, we prove constructively that secure FSSigs exist if a family of claw-free pairs of permutations (not necessarily with trapdoor) exist. We call the resulting scheme 'hiding scheme'.

In Chapter 4, we mention efficiency improvements for specific claw-free pairs.

Chapter 5 is a brief outlook, including applications.

1.4 Notation

Most of the notation is similar to [GoMR_88]: If a is a probabilistic algorithm, $a(I)$ denotes the probability space defined by running a on input I. For any probability space S, $P_S(X)$ denotes the probability of X; and $[S]$ the set of elements with positive probability.

$P(p(X_1, X_2,\ldots) \setminus X_1 \leftarrow a_1(I_1); X_2 \leftarrow a_2(I_2); \ldots)$ denotes the probability that the predicate $p(X_1, X_2,\ldots)$ is true after first a_1 on input I_1 has been executed and its result assigned to X_1, then a_2 etc.

DPA and PPA denote deterministic and probabilistic polynomial-time algorithms, resp.

The characteristic function of a predicate *pred* is denoted by 1_{pred}, string concatenation by &, string length by $|\bullet|$, and the unary representation of k by 1^k.

Signers will always be called Alice, recipients Bob.

2 Security of Fail-stop Signature Schemes

In FSSig schemes, we distinguish whether just one recipient Bob bears the risk of proved forgeries (2-party-case) or many participants (n-party case), since these recipients must participate in the key generation. Of course, in both cases, everybody can test signatures and proofs of forgery; in particular, judges. The 2-party case is quite common, e.g. in certain payment systems, or for personal promises. The FSSigs can be simpler in this case, since we need not exclude that Bob himself can compute proofs of forgery for Alice's signatures (since he would harm only himself if he did so).

In Sect. 2.1, we present FSSigs for the 2-party case where one part of the key generation has been left out. These objects are called basic FSSig schemes. In Sect. 2.2, we show that the missing part can be filled in for every basic FSSig scheme. In Sect. 2.3, we show how to construct an n-party FSSig scheme from every basic FSSig scheme. Finally we sketch that n-party FSSig schemes are secure in the sense of conventional signature schemes (Sect. 2.4.2); this is based on a proof of a corresponding weaker statement for basic FSSig schemes (Sect. 2.4.1).

2.1 Basic fail-stop signature schemes

A FSSig scheme must obviously contain the usual algorithms

- *sign* for signing and
- *test* for testing signatures,

and additional algorithms

- *proof*, used by Alice to construct a proof of forgery from a forgery, and
- *pr_test*, used by everybody to test if something is a proof of forgery.

Alice uses an algorithm

- G_A to generate pairs (*SK*, *PK*) of a secret and a public key.

SK is used by *sign* and *proof*, whereas *test* and *pr_test* work with *PK*.

Since Bob should be computationally secure that Alice cannot compute a proof of forgery, at least *pr_test* must have a one-way property holding against Alice. Thus we let Bob choose the corresponding part of the key. It has proved fruitful to mix the key choices as follows: First Bob uses a generator

- G_B to choose a **prekey** K, dependent on a cryptographical security parameter k.

Then G_A uses K as one of its inputs. Therefore, an incorrect choice of K may influence Alice's security, too. Hence Alice must test K. Sometimes it is difficult to test if K is a possible output of G_B (note that Bob must not reveal the internal choices of G_B), and there exists a superset $\mathcal{K}$ of the possible outputs for which membership is easier to test, and where $K \in \mathcal{K}$ still ensures Alice's security.

The membership test may have to be interactive, and we will first omit it from the definition. Later, a FSSig scheme will be a basic FSSig scheme augmented by this test (cf. Sect. 2.2 and Ch. 4).

There is a second security parameter σ for Alice's information-theoretical security; and, like in [GoMR_88], the number N of messages which can be signed must be fixed before choosing (*SK*, *PK*).

DEFINITION 1: A **basic FSSig scheme** for a message space $\mathcal{M} \subseteq \{0, 1\}^+$ is a 7-tuple (G_B, G_A, *sign*, *test*, *proof*, *pr_test*, $\mathcal{K}$), where the generators G_B, G_A are PPAs, *sign*, *test*, *proof*, and *pr_test* are DPAs, the range of *test* and *pr_test* is {*ok*, *not_ok*}, and $\mathcal{K}$ is a set, such that:

a) Supersets: For all $k \in \mathbb{N}$: $[G_B(1^k)] \subseteq \mathcal{K}$.

b) Faultless case: For all $k, \sigma, N \in \mathbb{N}$ and all prekeys $K \in \mathcal{K}$: Each output of $G_A(1^k, 1^\sigma, 1^N, K)$ is a pair (*SK*, *PK*), and for all messages $M \in \mathcal{M}$ and message numbers $\mu \in \{1,\ldots,N\}$:

$$S = sign(K, SK, M, \mu) \;\Rightarrow\; test(K, PK, M, \mu, S) = ok. \;\blacklozenge$$

The three parameters k, σ, and N will always be abbreviated as ***par*** $:= (1^k, 1^\sigma, 1^N)$.

Bob's security is lost if he has received a signature, and later it becomes invalid because a proof of forgery is shown. In our schemes, the proofs of forgery are independent of the messages. This is appropriate, since if any forgery occurred, the cryptographical assumptions have been wrong, and it is better to stop the scheme altogether, or at least to increase k. (This can be changed, cf. Ch. 5.) Also note that for Bob's security, one cannot assume that Alice chooses her keys correctly. She might even choose *PK* and *Proof* together.

DEFINITION 2: A basic FSSig scheme is **secure for the recipient (Bob)** iff for all PPAs $\mathcal{A}$ (representing an attacking Alice) and all $c \in \mathbb{N}$: For all sufficiently large k's,

$$\mathrm{P}(pr\_test(K, PK, Proof) = ok \setminus K \leftarrow G_B(1^k);\ (PK, Proof) \leftarrow \mathcal{A}(1^k, K)) < k^{-c}. \;\blacklozenge$$

Before defining Alice's security, we introduce some notation. Now we only assume $K \in \mathcal{K}$, which Alice tests, whereas in Bob's security, we could assume that Bob chooses K correctly.

DEFINITION 3: Assume a basic FSSig scheme, $K \in \mathcal{K}$, and $(SK, PK) \in G_A(par, K)$ have been chosen.

a) The set of possible **histories** is

$$\mathcal{Hist}(K, SK) := \{((M_1, S_1), \ldots, (M_{N'}, S_{N'})) \mid N' \le N;\ M_\mu \in \mathcal{M} \wedge S_\mu = sign(K, SK, M_\mu, \mu) \text{ for } \mu = 1, \ldots, N'\}.$$

b) The set of possible secret keys after a history is

$$\mathcal{SK}(K, PK, Hist) := \{SK \mid (SK, PK) \in [G_A(par, K)] \wedge Hist \in \mathcal{Hist}(K, SK)\}.$$

c) The set of **successful forgeries** after a history $Hist$ is that of new signatures, i.e.

$$\mathcal{Forg}(K, PK, Hist) := \{F = (M, \mu, S) \mid M \in \mathcal{M}, \mu \le N; \text{ and } (\mu > N' \vee M \ne M_\mu) \wedge test(K, PK, M, \mu, S) = ok\}.$$

d) A value F is a **provable forgery** after a history $Hist$ if

$$prov(K, SK, PK, Hist, F) :\Leftrightarrow pr\_test(K, PK, proof(K, SK, Hist, F)) = ok. \ \blacklozenge$$

DEFINITION 4: A basic FSSig scheme is **secure for the signer (Alice)** iff $\forall k, \sigma, N \in \mathbb{N}$, $\forall K \in \mathcal{K}$, $\forall (SK, PK) \in [G_A(par, K)]$, and $\forall Hist \in \mathcal{Hist}(K, SK)$:

For every successful forgery F, the conditional probability that Alice can compute a proof of forgery, given everything an attacker may know, is at least $1\text{-}2^{-\sigma}$. Formally: In the probability space defined by the assignment $(SK^*, PK^*) \leftarrow G_A(par, K)$, the requirement is

$$1\text{-}2^{-\sigma} \le \mathrm{P}(prov(K, SK^*, PK, Hist, F) \mid PK^* = PK \wedge Hist \in \mathcal{Hist}(K, SK^*)) \ \blacklozenge$$

REMARK: The last formula means

$$1\text{-}2^{-\sigma} \le \frac{\sum_{SK^* \in \mathcal{SK}(K,PK,Hist)} \mathrm{P}_{G_A(par,K)}(SK^*, PK) \bullet 1_{prov(K,SK^*,PK,Hist,F)}}{\sum_{SK^* \in \mathcal{SK}(K,PK,Hist)} \mathrm{P}_{G_A(par,K)}(SK^*, PK)}$$

DEFINITION 5: A **secure basic FSSig scheme** is one which is secure for both the signer and the recipient. ♦

2.2 Prekey tests

To see that it made sense to omit the prekey test from the definition of a basic FSSig scheme, we sketch two methods to provide any basic FSSig scheme with such a test.

REMARK: Neither of these methods needs the supersets. However, in the scheme which we propose in practice, a far more efficient prekey test can be used which does need them; see Ch. 4.

Cut-and-choose: The easiest method is a usual cut-and-choose technique [Rabi_78]. However, it makes signing and testing more inefficient. Bob proposes $2x$ prekeys instead of one, and Alice selects x ones for which Bob must reveal the construction. The remaining x prekeys are used. Alice chooses one key pair for each prekey, and signatures consist of x parts. However, a proof of forgery for one prekey suffices. This ensures Alice's security unless all x prekeys are wrong. The probability of this is exponentially small in x.

ZKP: The other method is to use a zero-knowledge proof (ZKP) system for $\mathcal{K}$. It must be correct in the information-theoretical sense, whereas it only needs to be computationally zero-knowledge. This guarantees that Alice is information-theoretical secure that the prekey is correct, but that the information she receives does not help her in computing false proofs of forgeries. (The exponentially small error probability can be subsumed in the error probability which Alice must accept anyway.)

Such ZKPs need not exist for some set $\mathcal{K}$. However, it is easy to see that any basis FSSig scheme stays secure if $\mathcal{K}$ is replaced by the union of the sets $[G_B(1^k)]$. This is in NP; and there are such ZKPs for all languages in NP [GoM1_86]. Only note that this system is again based on a key which both participants must trust. Thus, an additional cut-and-choose technique must be used for this key within the ZKP.

2.3 The *n*-party case

We show two general methods which transform each basic FSSig scheme into an n-party FSSig scheme. The first is conceptually simple, but makes signing and testing more inefficient. The second has a precomputation phase which might still be impracticable.

2.3.1 Many prekeys

One method is that each participant contributes one prekey. Alice chooses one key pair for each prekey (including her own). Signatures and, this time, also proofs of forgery consist of one part per prekey.

Alice's security: From the security of the basic FSSig scheme, it can easily be derived that for each successful forgery, the probability that Alice can compute a proof of forgery is at least $(1\text{-}2^{-\sigma})^n$.

Since $(1\text{-}2^{-\sigma})^n > 1\text{-}n\bullet 2^{-\sigma}$, one can choose $\sigma > \tau+\mathrm{ld}n$, if one wants the probability of unprovable forgery to be smaller than $2^{-\tau}$. Thus the overhead of the algorithms is polynomial in τ and n.

Security of a recipient Bob: For simplicity, we assume that Bob contributes the prekey K_1. Thus we can assume that K_1 is chosen correctly, whereas everything else can be wrong. Thus his security can be formalized as: For all PPAs $\mathcal{A}$ and all $c \in \mathbb{N}$, for all sufficiently large k

$$\mathrm{P}(pr\_test(K_1,\ldots,K_n, PK_1,\ldots,PK_n, Proof) = ok \setminus \\ K_1 \leftarrow G_B(1^k);\ (K_2,\ldots,K_n, PK_1,\ldots,PK_n, Proof) \leftarrow \mathcal{A}(1^k, 1^n, K_1)) < k^{-c}.$$

Assume the contrary. This trivially yields an algorithm $\mathcal{A}^*$ which finds proofs of forgery for the basic FSSig scheme with the same probability: $\mathcal{A}^*$ on input $(1^k, K_1)$ performs $\mathcal{A}$ on input $(1^k, 1^n, K_1)$ and outputs the first component of *Proof*. As long as n only grows polynomially with k, $\mathcal{A}^*$ is a PPA. This contradicts the security of the basic FSSig scheme.

2.3.2 Common choice of the prekey

In the second method, the participants agree on one prekey K.

Protocol: Starting point is the multi-party computation protocol of [ChDG_88], with a bit commitment scheme unconditionally secure for the verifier. Let C' be a circuit for $G_B(1^k)$, without private inputs, but with one random input ra. Since we have no trusted source of random bits, let C'' be a circuit which computes ra as the exor of n private inputs $ip_1,\ldots,ip_n$. The protocol of [ChDG_88] is applied to C, the composition of C'' and C'. If a participant disrupts the computation by submitting wrong messages, this is recognized with exponentially small error probability, and the computation is repeated without him.

Security Proof (Sketch): According to [ChDG_88], the output is unconditionally correct, i.e., an element of $[G_B(1^k)]$, with exponentially small error probability. This ensures Alice's security.

Now consider a recipient Bob: On the computational assumptions, the attackers know nothing about his input when they fix their own inputs. Thus the exor *ra* is completely random. Hence, in each complete protocol run, K is chosen according to the correct distribution. However, attackers might learn the result K early and disrupt the computation dependent on K. Their influence on the distribution of the final K is bounded by letting them choose among the K's from n complete protocol runs. One easily sees that if they could compute proofs of forgery with significant probability for these K's, and n is polynomial in k, this would contradict Def. 2.

REMARK: Usually, e.g., in [GMW_87, GaHY_88], such problems are avoided by enabling the other participants to simulate a disturbing participant. Then, however, one must assume that less than $n/2$ participants are attacking.

Finally, the zero-knowledge property of the protocol from [ChDG_88] implies that the intermediate information from the protocol run doesn't help the attackers in computing proofs of forgery. ♦

2.4 Relation to conventional signatures: security against forgery

A conventional signature scheme is called secure if, in the complexity-theoretic sense, existential forgery is impossible even after an adaptive chosen message attack. In Sect. 2.4.2, we show that this also holds for n-party FSSigs. This has a practical meaning, too: Of course, if forgeries could occur, Alice could prove them; however, it might be annoying to have them often. Now we are sure that, on computational assumptions, no forgeries occur at all.

We will first prove a corresponding statement for the 2-party case. However, in this case, Bob himself may be able to forge signatures. Thus we only prove that basic FSSig schemes are secure against existential forgery by outsiders, even after an adaptive chosen message attack.

2.4.1 2-party case

Like in [GoMR_88], an attacker who may use an adaptive chosen message attack is modelled by an interactive PPA $\mathcal{SR}$, the **signature requester**, and a normal PPA $\mathcal{SF}$, the **signature finder**.

DEFINITION 6: a) $\mathcal{Sign}$ denotes the signer as an interactive algorithm, i.e. $\mathcal{Sign}$ applies *sign* to messages communicated to it, keeping track of the current message number μ and the history *Hist*.

b) **comb**($\mathcal{SR}(I)$; $\mathcal{Sign}(J)$) denotes the combination of $\mathcal{SR}$ and $\mathcal{Sign}$, where I and J are private inputs, $\mathcal{SR}$ starts and determines the end, and the output is the view of $\mathcal{SR}$ (since that will be passed to $\mathcal{SF}$). The output will be written in the form (*Hist*, V), where *Hist* is a history and V the local variables of $\mathcal{SR}$. ♦

Note that this combination is a PPA on (I, J).

For the security against forgery, we can assume that both Bob and Alice choose their keys correctly. We must restrict the parameters σ and N to be polynomial in k, since they are part of the input for the outsider and would otherwise give even a polynomial algorithm time more than polynomial in k.

DEFINITION 7: A basic FSSig scheme is **secure against forgery**, iff for all interactive PPAs $\mathcal{SR}$ and all PPAs $\mathcal{SF}$, for all polynomials Q, Q', and all $c \in \mathbb{N}$: For all sufficiently large k's and $par(k) := (1^k, 1^{Q(k)}, 1^{Q'(k)})$,

$$k^{-c} > \quad P(F \in \mathcal{Forg}(K, PK, Hist) \setminus K \leftarrow G_B(1^k);\ (SK, PK) \leftarrow G_A(par, K);$$
$$(Hist, V) \leftarrow \mathrm{comb}(\mathcal{SR}(K, PK); \mathcal{Sign}(K, SK));\ F \leftarrow \mathcal{SF}(Hist, V)). \ ♦$$

THEOREM 1: A secure basic FSSig scheme is also secure against forgery. ♦

PROOF: Assume Alice's security and the contrary of Def. 7. Alice can use the adaptive chosen message attack to "forge" her own signatures, and apply *proof* to these "forgeries":

$\mathcal{A}$: On input $(1^k, K)$: $(SK, PK) \leftarrow G_A(par(k), K)$; $(Hist, V) \leftarrow \text{comb}(\mathcal{SR}(K,PK); \mathcal{Sign}(K,SK))$; $F \leftarrow \mathcal{SF}(Hist, V)$; $Proof := proof(K, SK, Hist, F)$; Output: $(PK, Proof)$.

We show that the probability $P_{success}$ that $\mathcal{A}$ obtains a proof of forgery is so high that it contradicts Bob's security (Def. 2). The basic idea is that Bob, in combination with the real Alice, could also perform $\mathcal{A}$ up to the choice of F; thus Alice's security implies that *proof* finds a proof of forgery for F with high probability. Formally, the probabilities of the adaptive chosen message attack must be transformed into those appearing in Alice's security.

$$P_{success} = \mathrm{P}(pr\_test(K, PK, Proof) = ok \setminus K \leftarrow G_B(1^k);\ (PK, Proof) \leftarrow \mathcal{A}(1^k, K))$$

$$= \mathrm{P}(prov(K, SK, PK, Hist, F) \setminus K \leftarrow G_B(1^k);\ (SK, PK) \leftarrow G_A(par(k), K);\ (Hist, V) \leftarrow \text{comb}(\mathcal{SR}(K, PK);\mathcal{Sign}(K,SK));\ F \leftarrow \mathcal{SF}(Hist, V))$$

$$= \sum_K (\mathrm{P}_{G_B(1^k)}(K) \sum_{SK,PK} (\mathrm{P}_{G_A(par(k),K)}(SK, PK) \sum_{Hist,V} (\mathrm{P}_{\text{comb}(\mathcal{SR}(K,PK);\mathcal{Sign}(K,SK))}(Hist, V) \cdot \sum_F \mathrm{P}_{\mathcal{SF}(Hist,V)}(F) \cdot 1_{prov(K,SK,PK,Hist,F)})))$$

$$\geq \sum_K (\mathrm{P}_{G_B(1^k)}(K) \sum_{PK} \sum_{Hist,V} \sum_{F\in \mathcal{Forg}(K,PK,Hist)} (\mathrm{P}_{\mathcal{SF}(Hist,V)}(F) \cdot \sum_{SK\in \mathcal{SK}(K,PK,Hist)} \mathrm{P}_{\text{comb}(\mathcal{SR}(K,PK);\mathcal{Sign}(K,SK))}(Hist, V) \cdot \mathrm{P}_{G_A(par(k),K)}(SK, PK) \cdot 1_{prov(K,SK,PK,Hist,F)})))$$

All the values $\mathrm{P}_{\text{comb}(\mathcal{SR}(K,PK);\mathcal{Sign}(K,SK))}(Hist, V)$ for $SK \in \mathcal{SK}(K, PK, Hist)$ are equal, say to $P(K, PK, Hist, V)$, since the behaviour of $\mathcal{SR}$ is independent of SK, and that of $\mathcal{Sign}$ is fixed by the condition on SK, which means that $\mathcal{Sign}$ produces just these signatures if confronted with the messages in $Hist$. If we factor this value out of the last sum, the last sum is the numerator from Alice's security (Def. 4). Thus

$$P_{success} \geq \sum_K (\mathrm{P}_{G_B(1^k)}(K) \sum_{PK} \sum_{Hist,V} \sum_{F\in \mathcal{Forg}(K,PK,Hist)} (\mathrm{P}_{\mathcal{SF}(Hist,V)}(F) \cdot P(K, PK, Hist, V) \cdot (1\text{-}2^{-\sigma}) \sum_{SK\in \mathcal{SK}(K,PK,Hist)} \mathrm{P}_{G_A(par(k),K)}(SK, PK)))$$

Rearranging the sums into their original order, we obtain the formula for the success of the outsider:

$$P_{success} \geq (1\text{-}2^{-\sigma}) \cdot \sum_K \mathrm{P}_{G_B(1^k)}(K) \sum_{SK,PK} \mathrm{P}_{G_A(par(k),K)}(SK, PK) \cdot \sum_{Hist,V} \mathrm{P}_{\text{comb}(\mathcal{SR}(K,PK);\mathcal{Sign}(K,SK))}(Hist) \sum_{F\in \mathcal{Forg}(K,PK,Hist)} \mathrm{P}_{\mathcal{SF}(Hist,V)}(F),$$

thus $P_{success} \geq (1\text{-}2^{-\sigma}) \cdot k^{-c} \geq k^{-(c+1)}$ for infinitely many k. ♦

2.4.2 *n*-party case

We restrict ourselves to an n-party FSSig scheme according to Sect. 2.3.1. (A similar proof is possible for the scheme from Sect. 2.3.2.) Since a FSSig scheme has more parameters than the signature schemes proposed in [GoMR_84, GoMR_88], we must adapt the security definition slightly.

For simplicity, assume Alice chooses the prekey K_1. Thus we can assume that K_1 is chosen correctly, whereas all others may be wrong.

Algorithms and predicates in the n-party FSSig scheme are distinguished from those in the basic FSSig scheme by an asterisk. We assume that their definitions are clear from the informal description in Sect. 2.3.1. Values without indices denote those from the n-party case.

DEFINITION 8: A FSSig scheme is **secure against existential forgery after adaptive chosen message attacks**, iff for all interactive PPAs $\mathcal{SR}$, all PPAs $\mathcal{KF}$ (prekey finder) and $\mathcal{SF}$, for all polynomials Q, Q', and all $c \in \mathbb{N}$: For all sufficiently large k's and $par := (1^k, 1^{Q(k)}, 1^{Q'(k)})$,

$$k^{-c} > \quad P(F \in \mathcal{F}org^*(K, PK, Hist) \setminus K_1 \leftarrow G_B(1^k);\ (K_2,\ldots,K_n, V') \leftarrow \mathcal{KF}(par, K_1);$$
$$(SK, PK) \leftarrow G_A{}^*(par, K);$$
$$(Hist, V) \leftarrow \text{comb}(\mathcal{SR}(K, PK, V');\mathcal{S}ign^*(K, SK));\ F \leftarrow \mathcal{SF}(Hist, V)) \ \blacklozenge$$

THEOREM 2: The n-party case of a secure basic FSSig scheme is secure against existential forgery after adaptive chosen message attacks. ♦

PROOF (SKETCH): Assume the contrary for $\mathcal{SR}$, $\mathcal{KF}$ and $\mathcal{SF}$. The basic idea is that every forger is an outsider with respect to the partial signature scheme based on K_1. Thus one can construct algorithms $\mathcal{SR}_1$ and $\mathcal{SF}_1$ contradicting Theorem 1: $\mathcal{SR}_1$ uses $\mathcal{KF}$ to choose additional prekeys and simulates the signature scheme with respect to them when $\mathcal{SR}$ requests signatures; $\mathcal{SF}_1$ just uses the first component of the signature forged by $\mathcal{SF}$. ♦

3 The General Hiding Scheme

We will construct a basic FSSig scheme from any family of claw-free pairs of permutations, not necessarily with a trapdoor. According to Sect. 2.2, 2.3, it suffices to construct a basic FSSig scheme.

REMARK: Also the ZKPs needed in Sect. 2.3.2 exist on this assumption.

Like in many other cryptographic schemes, ideas from LAMPORT's *one-time signatures* [DiHe_76, Merk_88] and MERKLE's tree-authentication [Merk_80, Merk_82] are used. However, they had to be modified to ensure both cryptographical security and proofs of forgery. The use of an error-detecting code is an idea from [Merk1_82].

3.1 Claw-free pairs of permutations and other basic algorithms

In contrast to [GoMR_88], where the generating algorithms output complete function descriptions, it is more convenient (and more realistic) to work with explicit keys here. Wherever an algorithm f depends on a key K, K will be the first parameter of f, and we will write $f^K(\bullet)$ instead of $f(K, \bullet)$.

Finally, we have to introduce the supersets needed for the FSSig scheme. Thus the definition of claw-free pairs takes the following form:

DEFINITION 9: A family of **claw-free pairs of permutations** with supersets is a quintuple $(G, d, f_0, f_1, \mathcal{K})$, where G and d are PPAs, f_0 and f_1 are DPAs, $\mathcal{K}$ is a set, and:

a) $\forall k \in \mathbb{N}$: $[G(1^k)] \subseteq \mathcal{K}$.
b) $\forall K \in \mathcal{K}$: The outputs of $d^K(\ )$ are uniformly distributed in their range D^K. $f_0{}^K$ and $f_1{}^K$ are permutations on D^K.
c) $\forall K \in [G(1^k)]$: $f_0{}^K$ and $f_1{}^K$ stop with error messages for inputs outside D^K.
d) For all PPAs $\mathcal{A}$ and all $c \in \mathbb{N}$ for all sufficiently large k's:

$$P(f_0{}^K(x_0) = f_1{}^K(x_1) \setminus K \leftarrow G(1^k);\ \ (x_0, x_1) \leftarrow \mathcal{A}(1^k, K)) \ < \ k^{-c}. \ \blacklozenge$$

REMARKS: Of course, if both $f_0{}^K$ on input x_0 and $f_1{}^K$ on input x_1 stop with an error message, this does not count as a claw.

The supersets have been used in b) and the normal sets in c) and d), since b) will be needed for Alice's security, and c) and d) for Bob's.

The following construction can also be adapted to claw-free families of permutations larger than pairs [Damg_88].

In the following, we will assume that such a family $(G, d, f_0, f_1, \mathcal{K})$ is given.

DEFINITION 10: A DPA *c_pr* and pairs of DPAs (*c_fix*, L') and (*c_ed*, L) are called **prefix-free encoding**, fixed-length encoding, and **error-detecting code**, resp., **for the claw-free pairs**, if for all $K \in \mathcal{K}$:

a) c_pr^K is a prefix-free encoding of D^K into $\{0, 1\}^+$,

b) $L'(K) \in \mathbb{N}$, and c_fix^K is an injective function from D^K into $\{0, 1\}^{L'(K)}$,

c) $L(K) \in \mathbb{N}$, and c_ed^K is a code from D^K into $\{0, 1\}^{L(K)}$ detecting all changes of 1's into 0's. ♦

Lemma 1: Encodings *c_pr*, (*c_fix*, L') and (*c_ed*, L) exist for each family $(G, d, f_0, f_1, \mathcal{K})$. ♦

PROOF: a) As *c_pr*, one can use the restriction of any prefix-free encoding *c_pr** of $\{0, 1\}^+$. An efficient *c_pr**, e.g., prefixes each message with its length, coded in fields of fixed length with an extension bit indicating if another length field follows.

b) The run-time of d is bounded by a polynomial Q; thus $|x| \le Q(|K|)$ for all $x \in D^K$. *c_fix* on input (K, x) computes $\lambda := Q(|K|)$, fills x with leading zeroes to length λ, and prefixes it with the original length in a field of length $\lambda' := \lfloor \mathrm{ld}(\lambda) \rfloor + 1$. $L'(K)$ is $\lambda + \lambda'$.

c) *c_ed* on input (K, x): $\Lambda := L'(K)$; $y := c\_fix^K(x)$; count the number z of zeroes in y; prefix x by z in binary encoding in a field of length $\Lambda' := \lfloor \mathrm{ld}(\Lambda) \rfloor + 1$. $L(K)$ is $\Lambda + \Lambda'$.

It is easy to see that all these algorithms are indeed DPAs with the necessary properties. In particular, changing a 1 into a 0 in the x-field of $c\_ed^K(x)$ implies an increase in z, and therefore a change from 1 to 0 there, and vice versa. ♦

In the following, we will also assume that ***c_pr***, (***c_ed***, ***L***), and a prefix-free encoding $\boldsymbol{c\_pr}_{\mathcal{M}}$ of a message space $\mathcal{M}$ are given.

REMARK: We can assume that each codeword of *c_ed* contains at least one 1.

To see that the hiding system is polynomial, note that the value $L(K)$, and not just its length, is polynomial in K (since it is the length of the output of *c_ed* on input (K, x) with $x \in D^K$).

3.2 Hiding functions and other collision-free functions

DEFINITION 11: From the given claw-free pairs and codes, we derive DPAs f, g, h, and *fold*. The second parameter of f is written as a lower index.

$$\begin{array}{lll}
\boldsymbol{f}^K{}_\xi(\chi) & := f_{\beta_1}{}^K(f_{\beta_2}{}^K(\ldots(f_{\beta_\sigma}{}^K(\chi))\ldots)) & \text{for } \xi = (\beta_1,\ldots,\beta_\sigma) \in \{0, 1\}^+. \\
\boldsymbol{g}^{K'}(x) & := f^K{}_\xi(\chi) & \text{for } K' = (1^\sigma, K),\ x = (\xi, \chi) \text{ with } \xi \in \{0, 1\}^\sigma. \\
\boldsymbol{h}^{K^*}(M) & := f^K{}_{c_pr_{\mathcal{M}}(M)}(\chi) & \text{for } K^* = (K, \chi) \text{ with } \chi \in D^K \text{ and } M \in \mathcal{M}. \\
\boldsymbol{fold}^K(x_1,\ldots,x_L) & := f^K{}_\xi(x_L) & \text{with } \xi := c_pr^K(x_1)\&\ldots\&c_pr^K(x_{L-1}).
\end{array}$$

Thus

$$\begin{array}{lll}
f^K{}_\xi: & D^K & \rightarrow D^K, \\
g^K: & \{0, 1\}^\sigma \times D^K & \rightarrow D^K, \\
h^{K^*}: & \mathcal{M} & \rightarrow D^K,
\end{array}$$

and the restriction of $fold^K$ to L arguments is a function

$$fold^K: \quad (D^K)^L \quad \rightarrow D^K. \ \blacklozenge$$

REMARKS: One could have made a function family (with a key generator etc.) of each of these DPAs, each with its own keys K, K', or K^*, resp., and prove properties of these families. However, in the hiding scheme we only need that g has a certain bundling property, and that all functions together are collision-free on one common key.

As function families, f corresponds to functions f_i in [GoMR_88], and h to the hash-functions in [Damg_88]. g is the central function for the hiding scheme; as a function family it could be called **hiding function**, since it can be used as a commitment scheme hiding parameter ξ unconditionally. The functions *fold* are more efficient hash functions for the tree authentication.

Lemma 2: a) $\forall K \in \mathcal{K}, \xi \in \{0, 1\}^+$: $f^K{}_\xi$ is a permutation on D^K.
b) $\forall K' = (1^\sigma, K)$ with $K \in \mathcal{K}$: Each $y \in g^{K'}(D^K)$ has 2^σ preimages. ♦

PROOF: a) Obvious, since all the functions $f_{\beta_i}{}^K$ are permutations.
b) We know $y \in D^K$. Because of a), for each $\xi \in \{0, 1\}^\sigma$, there is one $\chi \in D^K$ such that $y = f^K{}_\xi(\chi) = g^{K'}(x)$ for $x = (\xi, \chi)$. Since there are 2^σ choices for ξ, these are 2^σ preimages. ♦

The following algorithms transform g-, h-, and *fold*-collisions into claws of the claw-free pair for correctly chosen K. (For every function ϕ, a **ϕ-collision** is defined as a pair (x_0, x_1) with $x_0 \neq x_1$ and $\phi(x_0) = \phi(x_1)$.)

DEFINITION 12: We define DPAs $coll_f$, $coll_g$, $coll_h$, $coll_{fold}$:

coll$_f$: On input $(K, \xi_0, \chi_0, \xi_1, \chi_1)$: Let ξ^* be the common prefix of ξ_0 and ξ_1. If $\xi^* = \xi_0$ or $\xi^* = \xi_1$, abort. Now assume $\xi_0 = \xi^*\&0\&\xi'_0$ and $\xi_1 = \xi^*\&1\&\xi'_1$ (otherwise exchange 0 and 1). Then let $x_0 := f^K{}_{\xi 0}(\chi_0)$; $x_1 := f^K{}_{\xi 1}(\chi_1)$. Output: (x_0, x_1).

coll$_g$: On input $(K, x_0{}^*, x_1{}^*)$ with $x_i{}^* = (\xi_i, \chi_i)$: Apply $coll_f{}^K(\xi_0, \chi_0, \xi_1, \chi_1)$.

coll$_h$: On input (K^*, M_0, M_1) with $K^* = (K, \chi)$: Apply $coll_f{}^K(c\_pr_{\mathcal{M}}(M_0), \chi, c\_pr_{\mathcal{M}}(M_1), \chi)$.

coll$_{fold}$: On input (K, X_0, X_1) with $X_i = (x_1{}^{(i)}, \ldots, x_L{}^{(i)})$: Let $\xi_i := c\_pr^K(x_1{}^{(i)})\&\ldots\&c\_pr^K(x_{L\text{-}1}{}^{(i)})$. Apply $coll_f{}^K(\xi_0, x_L{}^{(0)}, \xi_1, x_L{}^{(1)})$. ♦

Lemma 3: For all $K \in [G(1^k)]$:
a) The output of $coll_f$ is a claw, i.e. $f_0{}^K(x_0) = f_1{}^K(x_1)$, whenever the input is a collision, i.e. $f^K{}_{\xi 0}(\chi_0) = f^K{}_\xi(\chi_1)$, and neither of ξ_0 and ξ_1 is a prefix of the other.
b) The output of $coll_g$, $coll_h$, or $coll_{fold}$ is a claw whenever the input is a collision. ♦

PROOF:
a) First, since neither of ξ_0 and ξ_1 is a prefix of the other, $\xi^* \neq \xi_0$ or $\xi^* \neq \xi_1$. Secondly, we have

$$f_{\xi^*}{}^K(f_0{}^K(f_{\xi'_0}{}^K(\chi_0))) = f_{\xi^*}{}^K(f_1{}^K(f_{\xi'_1}{}^K(\chi_1))).$$

Since $f_{\xi^*}{}^K$ is a permutation, this yields

$$f_0{}^K(f_{\xi'_0}{}^K(\chi_0)) = f_1{}^K(f_{\xi'_1}{}^K(\chi_1)) \;\;\Rightarrow\;\; f_0{}^K(x_0) = f_1{}^K(x_1).$$

b) One easily sees that in all three cases, $f_{\xi_0}{}^K(\chi_0) = f_{\xi_1}{}^K(\chi_1)$ holds for the inputs $(K, \xi_0, \chi_0, \xi_1, \chi_1)$ to $coll_f$. Thus we only need to check that neither of ξ_0 and ξ_1 is a prefix of the other.

g: Since ξ_0 and ξ_1 are both of length σ, we only need to show $\xi_0 \neq \xi_1$. $\xi_0 = \xi_1$ and $f_{\xi_0}{}^K(\chi_0) = f_{\xi_1}{}^K(\chi_1)$ would imply $\chi_0 = \chi_1$, since $f_{\xi_0}{}^K$ is a permutation, in contradiction to $x_0{}^* \neq x_1{}^*$.

h: By definition of *c_pr1*.

fold. The prefix-freeness of *c_pr* implies that of the concatenations. Thus if one of ξ_0 and ξ_1 were a prefix of the other, it could only be $\xi_0 = \xi_1$. This contradicts $X_0 \neq X_1$ like in the proof for g. ♦

3.3 The hiding scheme

Notation for trees: Assume there is a fixed binary tree, *tree*, of depth T (i.e. one with 2^T leaves). We denote its node values as follows: $node_1$ is the (value of the) root; and the left and right child of $node_\mu$ are $node_{2\mu}$ and $node_{2\mu+1}$, resp. We also count the leaves from left to right: $leaf_\mu = node_{2^T+\mu-1}$.

The τ-th forefather of $leaf_\mu$, i.e. the one τ layers above $leaf_\mu$, is $foref_\tau(\mu) = node_{(2^T+\mu-1)\text{div}2^\tau}$. sib_μ is the sibling of $node_\mu$, i.e. $node_{\mu+1}$ or $node_{\mu-1}$ if μ is even or odd, resp. For tree authentication, we need the sequence $aunts_\mu$ of all the siblings of the forefathers of $node_\mu$, i.e. $aunts_\mu = (sib_{(2^T+\mu-1)\text{div}2^\tau})_{\tau=0,\ldots,T-1}$.

DEFINITION 13: The **hiding scheme** for $\mathcal{M}$ derived from $(G, d, f_0, f_1, \mathcal{K})$, c_pr, (c_ed, L), and $c\_pr_\mathcal{M}$ consists of the same $\mathcal{K}$ and the following algorithms:

G_B = G.

***G_A*:** On input (par, K): Let $L := L(K)$; $K' := (1^\sigma, K)$; $\chi \leftarrow d^K(\,)$.

Preimages: Choose a matrix $r = (r_{\mu,\lambda})$ of preimages for the one-time signatures:

For $\mu = 1,\ldots,N$; $\lambda = 1,\ldots,L$:

Choose $\xi_{\mu,\lambda} \in \{0, 1\}^\sigma$ randomly; $\chi_{\mu,\lambda} \leftarrow d^K(\,)$; $r_{\mu,\lambda} := (\xi_{\mu,\lambda}, \chi_{\mu,\lambda})$.

Images: Compute the images:

$s_{\mu,\lambda} := g^{K'}(r_{\mu,\lambda})$ for $\mu = 1,\ldots,N$ and $\lambda = 1,\ldots,L$; $s := (s_{\mu,\lambda})$.

Authentication values: Compute the authentication values for complete signatures:

$leaf_\mu := fold^K(s_{\mu,1},\ldots,s_{\mu,L})$ for $\mu = 1,\ldots,N$.

Tree: Form a binary authentication tree, *tree*, of depth T, where T minimal with $2^T \geq N$: Choose arbitrary elements of D^K as $leaf_\mu$ for $\mu > N$, and let

$node_\mu := fold^K(node_{2\mu}, node_{2\mu+1})$ for $\mu = 2^T-1, 2^T-2,\ldots,1$.

Keys: The public key is the root of this tree, and the secret key consists of all information:

$PK := (k, \sigma, T, \chi, node_1)$; $SK := (k, \sigma, T, \chi, r, s, tree)$.

Output: (PK, SK).

***sign*:** On input (K, SK, M, μ) where $SK = (k, \sigma, T, \chi, r, s, tree)$: Let $K^* := (K, \chi)$.

Hash the message and apply the error-detecting code:

$M^* := c\_ed^K(h^{K^*}(M))$.

Let $(b_1,\ldots,b_L)$ be the bits of M^*. (By Def. 10c, L is the length of M^*.) Now the suitable preimages $r_{\mu,\lambda}$ are selected. For the other λ's, the images are needed for tree authentication:

$S' := (S_1,\ldots,S_L)$ where $S_\lambda := r_{\mu,\lambda}$ if $b_\lambda=1$, and $S_\lambda := s_{\mu,\lambda}$ if $b_\lambda=0$.

Additionaly, for tree authentication we need $S'' := aunts_\mu$.

Output: $S := (S', S'')$.

***test*:** On input (K, PK, M, μ, S): If, in the following, a value has the wrong number of components, or a subalgorithm stops with an error, *test* stops with the output *not_ok*.

Divide PK into $(k, \sigma, T, \chi, PK')$. Let $K^* := (K, \chi)$, $L := L(K)$, $K' := (1^\sigma, K)$, and $(b_1,\ldots,b_L) := M^* := c\_ed^K(h^{K^*}(M))$ as above. Divide S into (S', S''), S' into $(S'_1,\ldots,S'_L)$, and compute the values s^*_λ which should be $s_{\mu,\lambda}$, and the authentication value of the complete signature:

$s^*_\lambda := g^{K'}(S'_\lambda)$ if $b_\lambda=1$, and $s^*_\lambda := S'_\lambda$ if $b_\lambda=0$.

$foref^*_0 := fold^K(s^*_1,\ldots,s^*_L)$.

Now test the tree authentication: Divide S'' into $(aunt_0,\ldots,aunt_{T-1})$, and for $\tau=1,\ldots,T$, let

$foref^*_\tau := fold^K(foref^*_{\tau-1}, aunt_{\tau-1})$ if $(2^T+\mu-1)\text{div}2^{\tau-1}$ even;

$foref^*_\tau := fold^K(aunt_{\tau-1}, foref^*_{\tau-1})$ if $(2^T+\mu-1)\text{div}2^{\tau-1}$ odd.

Output: *ok*, if $foref^*_T = PK'$, else *not_ok*.

***proof*:** On input $(K, SK, Hist, F)$ where $SK = (k, \sigma, T, \chi, r, s, tree)$, $F = (M, \mu, S)$, and $Hist = ((M_1, S_1),\ldots,(M_{N'}, S_{N'}))$:

First test if F is a forged signature: Let $PK := (k, \sigma, T, \chi, node_1)$. If $test(K, PK, M, \mu, S) \neq ok$ or ($\mu \leq N'$ and $M = M_\mu$), stop. Here, all intermediate values from the description of *test* are defined, too. Now a bottom-up search for a collision takes place:

***h*-collision:** If $\mu \leq N'$ and $h^{K^*}(M) = h^{K^*}(M_\mu)$, stop with the output $Proof := (M, M_\mu)$.

Test for failure: Choose λ with b_λ=1 and $S'_\lambda \neq r_{\mu,\lambda}$. If there isn't any, stop with the output *bad_luck*.

***g*-collision:** If $s^*_\lambda = s_{\mu,\lambda}$, stop with the output $Proof := (S'_\lambda, r_{\mu,\lambda})$.

***fold*-collision:** If $foref^*_0 = leaf_\mu$, stop with the output $Proof := ((s^*_1,\ldots,s^*_L), (s_{\mu,1},\ldots,s_{\mu,L}))$.

Otherwise, there must be a collision in the tree-authentication. Let

$\tau^* := \min\{\tau \mid foref^*_\tau = foref_\tau(\mu)\}$.

This τ^* exists, since $foref^*_T = PK' = foref_T(\mu)$; and $\tau^* \neq 0$ because of $foref^*_0 \neq leaf_\mu$. Let X_0 be the pair of children of $foref_\tau(\mu)$, and X_1 the corresponding pair from F, i.e.

$(X_0, X_1) := ((foref^*_{\tau^*-1}, aunt_{\tau^*-1}), (node_{\mu'}, node_{\mu'+1}))$ if μ' even;

$(X_0, X_1) := ((aunt_{\tau^*-1}, foref^*_{\tau^*-1}), (node_{\mu'-1}, node_{\mu'}))$ if μ' odd.

$Proof := (X_0, X_1)$.

***pr_test*:** On input $(K, PK, Proof)$ where $PK = (k, \sigma, T, \chi, PK')$: Let $K^* := (K, \chi)$.

Output: *ok*, if *Proof* is an *h*-, *g*-, or *fold*-collision; otherwise *not_ok*. ♦

THEOREM 3: A hiding scheme is a basic FSSig scheme. ♦

PROOF: Tedious but easy (cf. [Pfit1_89]). ♦

3.4 Security of the hiding scheme

THEOREM 4: A hiding scheme is secure for the recipient (Bob). ♦

PROOF: Assume the contrary for a PPA $\mathcal{A}$. Then, with Lemma 3b, one easily sees that the algorithm

$\mathcal{A}^*$: On input $(1^k, K)$: $(PK, Proof) \leftarrow \mathcal{A}(1^k, K)$; divide PK into $(k, \sigma, T, \chi, PK')$; let $K^* := (K, \chi)$;
test if $Proof = (x_0, x_1)$ and (x_0, x_1) is an *h*-, *g*-, or *fold*-collision;
Output $coll_h^{K^*}(x_0, x_1)$, $coll_g^{K}(x_0, x_1)$, or $coll_{fold}^{K}(x_0, x_1)$ accordingly.

contradicts Def. 9. ♦

THEOREM 5: A hiding scheme is secure for the signer (Alice). ♦

Lemma 4: If *pr_test* seems to find a proof of forgery, this is a proof of forgery indeed. Formally: Assume $K \in \mathcal{K}$, $(PK, SK) \in [G_A(par, K)]$ and $Hist \in \mathcal{Hist}(K, SK)$. If *proof* does not abort, then

$$Proof = proof(K, SK, Hist, F) \neq bad\_luck \Rightarrow pr\_test(K, PK, Proof) = ok. \quad ♦$$

PROOF: One easily sees that the things claimed to be *h*-, *g*-, and *fold*-collisions in the description of *proof* are such collisions indeed. ♦

Proof of Theorem 5: Let $K \in \mathcal{K}$, $(PK, SK) \in [G_A(par, K)]$, $Hist \in \mathcal{Hist}(K, SK)$, and $F = (M, \mu, S) \in \mathcal{Forg}(K, PK, Hist)$. Let $K^* := (K, \chi)$ where $PK = (k, \sigma, T, \chi, PK')$.

Case 1: The one-time signature is not new, i.e. $\mu \leq N'$ and $h^{K^*}(M) = h^{K^*}(M_\mu)$. From the description of *proof* it is clear that it has an output $Proof \neq bad\_luck$, independent of SK^*. Thus we can apply Lemma 4, and we see that the forgery is provable with probability 1.

Case 2: $\mu > N'$ or $h^{K^*}(M) \neq h^{K^*}(M_\mu)$. Informally, we will prove that, with probability at least $1\text{-}2^{-\sigma}$, the attacker chooses at least one preimage in F wrong:

a) First we fix a bit for which the attacker does not know the correct preimage yet: If $\mu > N'$, choose any λ with b_λ=1 (cf. the remark at the end of § 3.1). Otherwise $h^{K^*}(M) \neq h^{K^*}(M_\mu)$. Let $(b_1,\ldots,b_L)$ and $(b^*_1,\ldots,b^*_L)$ be the bit sequences of $c\_ed^K(h^{K^*}(M))$ and $c\_ed^K(h^{K^*}(M_\mu))$, resp. By definition of c_ed, there is at least one λ with

$$b_\lambda=1;\ b^*_\lambda=0.$$

b) Now we define secret keys as equivalent if their only difference is in this preimage:

$$SK^{(1)} \approx SK^{(2)} :\Leftrightarrow r^{(1)}_{\mu',\lambda'} = r^{(2)}_{\mu',\lambda'} \text{ for all } (\mu',\lambda')\neq(\mu,\lambda) \text{ and } s^{(1)}_{\mu,\lambda} = s^{(2)}_{\mu,\lambda}.$$

Each $s_{\mu,\lambda}$ has at least 2^σ g-preimages because of Lemma 2b. Hence each class modulo $\approx$ has at least 2^σ members.

c) One can easily see that $\mathcal{SK}(K, PK, Hist)$ consists of complete classes modulo $\approx$:

Let $SK^{(1)} \approx SK^{(2)}$ and $SK^{(1)} \in \mathcal{SK}(K, PK, Hist)$, i.e. $(SK^{(1)}, PK) \in [G_A(par)] \wedge Hist \in \mathcal{Hist}(K, SK^{(1)})$, i.e. if $Hist = ((M_1, S_1),\ldots, (M_{N'}, S_{N'}))$, then $S_{\mu'} = sign(K, SK^{(1)}, M_{\mu'}, \mu')$ for each μ'.

We must show that also $(SK^{(2)}, PK) \in [G_A(par)]$ and $S_{\mu'} = sign(K, SK^{(2)}, M_{\mu'}, \mu')$ for each μ'.

The first part is correct since in the description of G_A, PK depends on the matrix s of the secret key only. The same holds for the S''-part of each signature.

Next, for $\mu' \neq \mu$, $sign(K, SK^{(2)}, M_{\mu'}, \mu')$ only depends on line μ' of the matrix $r^{(2)}$, which is equal to that of $r^{(1)}$. Hence $S_{\mu'} = sign(K, SK^{(2)}, M_{\mu'}, \mu')$, too.

Finally, if $\mu \le N'$, then the S'-part of $sign(K, SK^{(2)}, M_\mu, \mu)$ is $(S^{(2)}_1,\ldots, S^{(2)}_L)$ where $S^{(2)}_{\lambda'} := r^{(2)}_{\mu,\lambda'}$ if $b^*_{\lambda'} = 1$, and $S^{(2)}_{\lambda'} := s^{(2)}_{\mu,\lambda'}$ if $b^*_{\lambda'} = 0$. For $\lambda' \neq \lambda$, both $r^{(2)}_{\mu,\lambda'} = r^{(1)}_{\mu,\lambda'}$ and $s^{(2)}_{\mu,\lambda'} = s^{(1)}_{\mu,\lambda'}$, and by a), $b^*_\lambda = 0$, thus $S^{(2)}_\lambda = S^{(1)}_\lambda$, too.

d) Let $\mathcal{C}$ be the set of these classes. Thus we obtain for the probability from Alice's security

$$\begin{aligned} P^* &:= \mathrm{P}(prov(K, SK^*, PK, Hist, F) \mid SK^* \in \mathcal{SK}(K, PK, Hist)) \\ &= \sum_{C\in\mathcal{C}} \mathrm{P}(prov(K, SK^*, PK, Hist, F) \mid SK^* \in C) \bullet \mathrm{P}(SK^* \in C \mid SK^* \in \mathcal{SK}(K, PK, Hist)) \end{aligned}$$

e) In each class $C \in \mathcal{C}$, there is at most one SK^* with $r_{\mu,\lambda}=S'_\lambda$. One can easily, though tediously, check that $(b_\lambda=1 \wedge r_{\mu,\lambda}\neq S'_\lambda)$ implies that $proof(K, SK^*, Hist, F)$ produces an output $Proof \neq bad\_luck$. With Lemma 4, this implies that C contains at most one SK^* for which $prov(K, SK^*, PK, Hist, F)$ may be false.

f) All secret keys are equally probable, since each one is fully determined by its matrix r, and all r's are equally probable since $d^K(\,)$ defines a uniform distribution. Thus for each C, the first factor in the last line of d) is

$$\frac{|\{SK^*\in C \mid prov(K,SK^*,PK,Hist,F)\}|}{|C|} \ge \frac{|C| - 1}{|C|} \ge 1 - 2^{-\sigma}.$$

Hence
$$P^* \ge (1 - 2^{-\sigma}) \sum_{C\in\mathcal{C}} \mathrm{P}(SK^* \in C \mid SK^* \in \mathcal{SK}(K, PK, Hist)) = 1 - 2^{-\sigma}. \ \blacklozenge$$

4 Special Hiding Schemes

Claw-free pairs based on factorization: The most efficient familiy of claw-free pairs of permutations known is the GMR-family, i.e. the family based on factoring from [GoMR_88]. In this case, each D^K

is a large subset of $\mathbb{Z}_K$. Thus one would not use *c_pr* from Lemma 1, but would use numbers of fixed length with leading zeroes directly. Then also *c_fix* is needed no longer in the Proof of Lemma 1, c).

Supersets from [BlPW_90]: In [BlPW_90], it has been proposed to use the GMR-family with all odd numbers as superset $\mathcal{K}$. Thus, instead of the expensive methods from Sect. 2.2, Alice only needs to test that K is odd. Thus the precomputation for these FSSigs is not much worse than for conventional signatures: In a first round, everybody publishes their prekeys, and in a second round, the public keys.

The statements proved in [BlPW_90] fit into the scheme proved here as follows: In general, f_0 and f_1 are no longer permutations on D^K. However, it was shown that $d^K(\ )$ still defines a uniform distribution on D^K, if it is suitably generalized to all odd numbers, and that Lemma 2b still holds. With this, the proof of Alice's security is still correct. The situation for correctly chosen K, and thus Bob's security, is unchanged.

Small message space: In an application where the message space $\mathcal{M}$ is small, the hash function can be omitted. Thus $\mathcal{M}$ would be considered small if messages are shorter than the elements of D^K for a suitable security parameter k. Signing then starts directly with the error-detecting code.

Claw-free pairs based on the discrete log: The (otherwise less efficient) collision-free pairs based on the discrete log from [Damg_88] have an advantage if the idea from Sect. 2.3.2 is used, because in that case, any protocol for choosing mutually trusted random bits should suffice.

5 Outlook

Efficiency: One lower bound on the efficiency of FSSig schemes different from that of conventional digital signature schemes, seems provable: For each new (complete) signature, σ new random bits are needed.

However, we do not see any theoretical reason why a FSSig scheme should necessarily be based on one-time signatures. It should be worthwhile looking for a scheme based on some sort of trapdoor, where signatures would have the same length, and signing and testing similar run-time, as in conventional signature schemes.

Applications: Important applications for the 2-party case are payment systems, where clients sign and only a bank, not other clients, must trust the signatures. If the bank itself uses conventional signatures, the clients are unconditionally secure. This is sensible, since the bank can usually choose the signature scheme and the security parameter. A particularly efficient application is possible in the payment system from [ChFN_90], see [Pfit1_89].

The n-party case has been used to obtain Byzantine Agreement protocols (BAPs) with new security features [WaPf_89]: a Fail-stop BAP, and an Adaptive BAP which works if either of the two usual assumptions (a computational assumption or one on the number of attackers) holds.

Undeniable signatures: Recently, so-called "undeniable" signatures have been investigated [ChAn_89, Chau1_90, BCDD_90]. Such a signature cannot be shown round by the recipient alone, but the signer cannot deny it in court. A scheme with similar properties, but unconditionally secure for the signer (and less efficient), can quite easily be derived from the 2-party case of the hiding scheme, if claw-free pairs with trapdoor [GoMR_88] are used: Bob chooses a different prekey K_μ for each (complete) signature and one, K_0, for tree authentication. A proof of forgery for K_μ invalidates just the μ-th signature. Bob publishes K_0 and a tree authentication value of all K_μ's, and tells Alice all K_μ's.

Alice's keys are defined as in Ch. 3, but she may tell Bob the images *s*. Now Bob can sign, too, but Alice can, with high probability, deny each signature made by Bob. So if Bob shows a signature that doesn't mean Alice has given it.

Acknowledgements

We are pleased to thank *Andreas Pfitzmann* for several good ideas, *Gerrit Bleumer* for insisting that we make things simpler, *Manfred Böttger*, *David Chaum*, and *Dirk Fox* for helpful discussions, and the German Science Foundation (*DFG*) for financial support.

Literature

BeMi_88 Mihir Bellare, Silvio Micali: How to sign given any trapdoor function; 20th STOC, ACM, New York 1988, 32-42.

BCDD_90 Joan Boyar, David Chaum, Ivan Damgård, Whitfield Diffie, Torben Pedersen: Convertible undeniable signatures; to be presented at Crypto '90.

BlPW_90 Gerrit Bleumer, Birgit Pfitzmann, Michael Waidner: A Remark on a Signature Scheme where Forgery can be Proved; Eurocrypt '90, Abstracts, 403-407.

ChAn_89 David Chaum, Hans van Antwerpen: Undeniable Signatures; Crypto '89, August 20-24 1989, Abstracts, 205-212.

Chau1_90 David Chaum: Zero-Knowledge Undeniable Signatures; Eurocrypt '90, Abstracts, 419-426.

ChDG_88 David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty Computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

ChFN_90 D. Chaum, A. Fiat, M. Naor: Untraceable Electronic Cash; Crypto '88, LNCS 403, Springer Verlag, Berlin 1990, 319-327.

ChRo_90 David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures; to be presented at Crypto '90.

Damg_88 Ivan Bjerre Damgård: Collision free hash functions and public key signature schemes; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 203-216.

DiHe_76 Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644-654.

GaHY_88 Zvi Galil, Stuart Haber, Moti Yung: Cryptographic computation: secure fault-tolerant protocols and the public-key model; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 135-155.

GiMS_74 E. N. Gilbert, F. J. Mac Williams, N. J. A. Sloane: Codes which detect deception; The Bell System Technical Journal 53/3 (1974) 405-424.

GMW_87 Oded Goldreich, Silvio Micali, Avi Wigderson: How to play any mental game - or - a completeness theorem for protocols with honest majority; 19th STOC, ACM, New York 1987, 218-229.

GoM1_86 Oded Goldreich, Silvio Micali, Avi Wigderson: Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design; 27th FOCS, IEEE Computer Society, 1986, 174-187.

GoMR_84 Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A "Paradoxical" Solution to the Signature Problem; 25th FOCS, IEEE Computer Society, 1984, 441-448.

GoMR_88 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.

Merk_80 Ralph C. Merkle: Protocols for Public Key Cryptosystems; Proceedings of the 1980 Symposium on Security and Privacy, April 14-16, 1980 Oakland, California, 122-134.

Merk_82 Ralph C. Merkle: Protocols for Public Key Cryptosystems; AAAS Selected Symposium 69, Secure Communications and Asymmetric Cryptosystems; Gustavus J. Simmons (ed.); Westview Press, Boulder, Colorado; 1982, 73-104.

Merk_88 Ralph C. Merkle: A digital signature based on a conventional encryption function; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 369-378.

Merk1_82 Ralph Charles Merkle: Secrecy, authentication, and public key systems; UMI Research Press 1982.

NaYu_89 Moni Naor, Moti Yung: Universal One–way Hash Functions and their Cryptographic Applications; 21st STOC, ACM, New York 1989, 33-43.

Pfit1_89 Birgit Pfitzmann: Für den Unterzeichner sichere digitale Signaturen und ihre Anwendung; Diplomarbeit, Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe 1989.

PfWa_91 Birgit Pfitzmann, Michael Waidner: Fail-stop-Signaturen und ihre Anwendung; accepted for VIS '91, will be published in LNCS, Springer-Verlag, Berlin 1991.

PfWa1_91 Birgit Pfitzmann, Michael Waidner: Fail-stop Signatures and their Application; accepted for Securicom 91, Paris, March 1991.

Rabi_78 Michael O. Rabin: Digitalized Signatures; Foundations of Secure Computation, ed. by R.A. DeMillo, D.P. Dobkin, A.K. Jones, R.J. Lipton; Academic Press, N.Y. 1978, 155-166.

RSA_78 R. L. Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; CACM 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96-99.

SaYu_90 Alfredo De Santis, Moti Yung: On the Design of Provably–Secure Cryptographic Hash Functions; Eurocrypt '90 – Abstracts, Scanticon, Århus, Denmark, May 1990, 377–397.

WaPf_89 Michael Waidner, Birgit Pfitzmann: Unconditional Sender and Recipient Untraceability in spite of Active Attacks – Some Remarks; Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 5/89, März 1989.