UNIVERSITÄT KARLSRUHE

FAKULTÄT FÜR INFORMATIK

Postfach 6980, W-7500 Karlsruhe 1, Fed. Rep. of Germany

Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer

David Chaum[1], Eugène van Heijst[1], Birgit Pfitzmann[2]

[1]Centre for Mathematics and Computer Science (CWI),
Kruislaan 413, NL-1098 SJ Amsterdam
[2]Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe

Interner Bericht 1/91
February 1991

Abstract

"Undeniable" (or perhaps rather "invisible") signatures are digital signatures which the recipient cannot show round without the help of the signer. If forced to either acknowledge or deny a signature, however, the signer cannot deny it if it is authentic.

We present the first undeniable signature scheme which is unconditionally secure for the signer (except for an exponentially small error probability). The security for the recipient is provably as secure as the discrete logarithm in certain groups. Besides, this is the first practical cryptographically strong undeniable signature scheme at all. In many cases, it is more efficient than previous signature schemes unconditionally secure for the signer.

Interesting subprotocols are efficient cryptographically collision-free hash functions based on the discrete log, and efficient perfectly hiding commitments on numbers modulo a prime with particular inequality proofs.

Table of contents

1 Introduction

Digital signatures in general: Digital signatures are intended to provide legal security in digital communication, as handwritten signatures should do in conventional communication: A recipient of a signed message should be able to prove to third parties, often a court, that he has indeed received this message, e.g. a promise, from a specific signer. On the other hand, it should not be possible to cheat signers by forging signatures.

Digital signatures were informally introduced in [DiHe_76], and a formal definition of what was thought to be optimal security for them was given in [GoMR_84, GoMR_88]. We will call signatures according to this definition conventional digital signatures.

Recently it has been discovered that this definition was not yet as general as possible, and that schemes with more (or different) security features are possible. Here we present one such scheme.

Conventional digital signatures: In conventional digital signature schemes, there is always a globally known test function *test*. Each signer Sibyl has a public key *PK*, and a value s is defined to be Sibyl's signature under a message m if $test(PK, m, s) = true$.

This implies that cheating signers is within NP, since one can guess a signature and test its correctness. Moreover, so far, the security of the signers always relies on unproved assumptions like "factoring integers with large prime factors is hard".

It also implies that if a recipient Rick has received a signature from Sibyl, he can show to everybody else that Sibyl signed this message, even if there is no dispute among them, and thus no real need for him to show the signature.

Undeniable signatures: Undeniable (or perhaps rather "invisible") signatures have been introduced in [ChAn_90] and further developed in [Chau1_90, BCDP_90]. They are digital signatures which provide more privacy: A recipient of a signature cannot show it without the help of the signer. This property will be called invisibility in the following. If, however, the signer is forced to either deny or acknowledge a valid signature, e.g. in court, she cannot deny it.

In the previous schemes, the recipients are unconditionally secure, except for an exponentially small error probability. The signers, however, (as with conventional digital signature schemes) can be cheated with forged signatures if a cryptographical assumption turns out to be wrong; i.e., in the efficient schemes, if discrete logarithms can be computed.

Moreover, none of the efficient undeniable signature schemes so far is cryptographically strong, i.e. proved to be as secure as the discrete log. (There is, however, an impractical cryptographically strong scheme in [BCDP_90] for demonstration purposes.) In particular, no security against active attacks (cf. [GoMR_88]) has been proved for the efficient schemes. (With the pure schemes, successful active attacks are even known; but of course, one can try to avoid them with redundancy predicates or hash functions as with conventional signature schemes such as RSA [RSA_78]. On the other hand, this property can be an advantage: Some of these schemes enable blind signatures, cf. [Cha8_85].)

Unconditional security for signers: Unconditional security for signers has already been considered as an additional, or at least alternative, security feature for normal "non-undeniable" (or "visible") signature schemes.

In [WaPf_89, Pfit1_89, BlPW_90, PfWa2_90], fail-stop signatures were introduced. They are cryptographically secure against forgeries in the sense of [GoMR_88]. In addition, if a forgery occurs nevertheless, the signer can prove this unconditionally to everyone, except for an exponentially small error probability. In particular, if one defines that a signature becomes invalid once a proof of forgery has been shown, signers are unconditionally secure, and the security of the recipients is cryptographically strong. The name "fail-stop" refers to the fact that as soon as the cryptographical assumption is broken, everybody notices this and the scheme can be stopped; hopefully, before too much damage has occurred.

Unconditional signatures, i.e. signature-like schemes where both parties are unconditionally secure, except for an exponentially small error probability, were introduced in [ChRo_90]. (The term "signature-like" was used because these signatures, in contrast to fail-stop signatures, are not conform with the specification of conventional signatures, i.e. they cannot just be used in any protocol using signatures without additional measures.)

However, so far, none of these schemes is efficient in all cases: Fail-stop signatures are efficient for one-bit messages, but they (i.e. their length) grow linearly with the length of the message (up to a certain point, since messages can be hashed). Unconditional signatures have a complicated precomputation phase and grow linearly with the number of possible recipients. Both kinds of signatures are even longer if signatures are to be transferred several times (although this is not a natural thing to do in applications where we might want undeniable signatures); for those of [ChRo_90], in the general case, it is not even known how long they must be.

The new scheme: We present the first undeniable signature scheme which is unconditionally secure for the signer, except for an exponentially small error probability. This is nice for two reasons:

- First, undeniable signatures make special precautions for the signer with respect to privacy, so it is good to take special care about the signer's security against fraud, too.
- Secondly, with all cryptographical protocols where just one of two parties is unconditionally secure, it is good to have both variants to choose from, such as with zero-knowledge proofs unconditionally secure for the prover or the verifier, or commitment schemes unconditionally hiding or committing.

Additionally, the scheme presented here is the first practical undeniable signature scheme where the security for the remaining party (i.e. the party who is not unconditionally secure), here the recipients, is cryptographically strong.

In many cases, it is also more efficient than previous schemes unconditionally secure for the signers. (It is, however, not a fail-stop scheme: If recipients are cheated, they cannot prove this to everybody else.)

Interesting subprotocols: Subprotocols that may be interesting in their own right are efficient cryptographically collision-free hash functions based on the discrete log (cf. Lemma 3 (§ 5.2) and § 7.3), and efficient perfectly hiding commitments on numbers modulo a prime, with particular inequality proofs (Lemmas 6, 8, and 9 (§ 6.2.3)).

Overview: We first present the basic ideas of the new signature scheme in an abstract way (§ 2). After some notation (§ 3), we present the structure of a simple case of the actual scheme, i.e. we assume that just one message of fixed length is to be signed (§ 4), and show its security (§ 5). In § 6, we fill the structure from § 4 with certain efficient zero-knowledge proofs. Finally, we extend the scheme to handle many messages of arbitrary length efficiently (§ 7).

2 Basic ideas

Our scheme combines ideas of previous undeniable signatures and of fail-stop signatures.

The basic idea to achieve the invisibility which characterizes undeniable signatures is that there is no test function for signatures, which everybody can evaluate himself. Instead, a signature can only be verified by an interactive protocol between the signer and the recipient, preferably in zero-knowledge.

The basic idea of fail-stop signatures is that, given a public key and perhaps previous signatures, each new message has many acceptable signatures s. On a cryptographical assumption, however, the signer can compute just one of them, say s^*. If someone forges a signature, with high probability he will not use s^*, since all acceptable signatures look equally likely to him. Thus there are now two different signatures for one message. This counts as a proof of forgery, since it shows that the cryptographical assumption has been broken.

In combination with invisibility, the latter idea is changed a bit: When the real signer receives the forged signature, say sf, she cannot prove that s^* and sf are two different signatures for the same message, since she cannot carry out the verification protocol for sf. Instead, she will only prove (in zero-knowledge) that her real signature is not sf. (However, this is no proof that the cryptographical assumption has been broken, since it can be done for any value sf. Thus this scheme does not have the fail-stop property of fail-stop signatures.)

3 Assumptions and notation

3.1 Probabilistic algorithms

We mainly use the usual notation for probabilistic algorithms, see e.g. [GoMR_88].

According to the reaction of some preliminary readers, lots of details are presented in one proof of cryptographical security (Lemma 3). For this, some basic facts about probabilistic algorithms are assembled in very great detail, too:

1. For each input I, a probabilistic algorithm $\mathcal{A}$ defines a probability space on the set of possible outputs; let

$$\mathrm{P}_{\mathcal{A}(I)}(O)$$

denote the probability of output O if $\mathcal{A}$ is run on input I.

Remark: The function assigning such a probability space to each input can be seen as the semantics of the probabilistic algorithm, i.e. a machine independent definition of its behaviour, just like the function computed by a deterministic algorithm.

Depending on the machine model, this probability space can be derived from more elementary ones. E.g., in the Turing machine scenario, a probabilistic algorithm is a deterministic Turing machine with an additional read only input tape, which is only read forwards. Also by definition, this tape is filled with random bits which are independent of any other random bits anywhere. The basic elementary events are the contents of the random tape, and the algorithm with a specific input is a random variable from that space into the space of possible outputs. There, it induces a second probability space, whose elementary events are the single outcomes, with the probabilities defined above.

2. For any predicate p on the space of possible outputs, its probability is

$$\mathrm{P}_{\mathcal{A}(I)}(p) = \sum_{O} \mathrm{P}_{\mathcal{A}(I)}(O) \bullet 1_{p(O)},$$

where $1_{p(O)}$ denotes the characteristic function. (The O is normally not written in the index, but since most predicates will be equations, this notation is more practical.) This probability is often written as $\mathrm{P}(p(O) \mid O \leftarrow \mathcal{A}(I))$. Here, however, the stroke "|" is reserved for conditional probabilities.

For any event E, i.e. set of possible outputs, the probability is

$$\mathrm{P}_{\mathcal{A}(I)}(E) = \sum_{O \in E} \mathrm{P}_{\mathcal{A}(I)}(O).$$

Remark: This is just the probability of composite events, expressed in terms of the probabilities of elementary events.

3. Let a probabilistic algorithm $\mathcal{B}$ be defined as the composition of probabilistic algorithms as follows:

$\mathcal{B}$: On input I:
$x_1 \leftarrow \mathcal{A}_1(f_0(I))$;
$x_2 \leftarrow \mathcal{A}_2(f_1(I, x_1))$;
…
$x_m \leftarrow \mathcal{A}_m(f_{m-1}(I, x_1, \ldots, x_{m-1}))$;
Output $O := f_m(I, x_1, \ldots, x_m)$.

Then we can derive the probabilities of the outputs of $\mathcal{B}$ in terms of probabilities of the algorithms $\mathcal{A}_i$ as follows: First we obtain the probability of a history of $\mathcal{B}$ on input I as

$$\mathrm{P}_{\mathcal{B}(I)}(\text{"History is } x_1,\ldots, x_m\text{"}) = \mathrm{P}_{\mathcal{A}_1(f_0(I))}(x_1) \bullet \ldots \bullet \mathrm{P}_{\mathcal{A}_m(f_{m-1}(I, x_1,\ldots, x_{m-1}))}(x_m).$$

For an output O, this implies

$$\mathrm{P}_{\mathcal{B}(I)}(O) = \sum_{x_1,\ldots,x_m} 1_{f_m(I, x_1, \ldots, x_m)=O} \bullet \mathrm{P}_{\mathcal{B}(I)}(\text{"History is } x_1,\ldots, x_m\text{"})$$

$$= \sum_{x_1} (\mathrm{P}_{\mathcal{A}_1(f_0(I))}(x_1) \bullet \sum_{x_2} (\ldots \bullet \sum_{x_m} (\mathrm{P}_{\mathcal{A}_m(f_{m-1}(I, x_1,\ldots, x_{m-1}))}(x_m) \bullet 1_{f_m(I, x_1,\ldots, x_m)=O})\ldots)).$$

In the following, the parantheses in this kind of formula will be omitted.

Remark: One can either prove this formula for a specific machine model, e.g. by going back to a definition of $\mathcal{B}$ as a deterministic algorithm on I and its random tape.

Or one can see the last formula as the definition of the semantics of the composition of probabilistic algorithms, independent of a machine model.

3.2 Special assumptions and notation

Our discrete log assumption: We need a group $\mathcal{G}_p$ of known prime order p where everyone can perform the group operations and choose random elements efficiently, but the discrete log is hard.

The security of both signers and recipients will depend on the primality of p. In [ChAn_90], there are three proposals for $\mathcal{G}_p$:

1. $\mathcal{G}_p = \mathrm{GF}(2^n)^*$ where $p = 2^n-1$ is a Mersenne-prime.
2. $\mathcal{G}_p = \mathbb{Z}_q^{\ *}/\{\pm 1\}$ where $q = 2p+1$ and both p and q are prime. $\mathcal{G}_p$ can be represented by $\{1, \ldots, p\}$.
3. Equivalently to 2., one can choose $\mathcal{G}_p = \mathrm{QR}_q$ (the quadratic residues).

Groups on elliptic curves are also possible (following ideas from [Mil1_86] or [Kobl1_90]).

More formally, our cryptographical assumption, needed to prove the security of the recipients, is that one of these is an infinite sequence of groups where the discrete log cannot be solved with significant probability in polynomial time.

Vector notation: For $g = (g_1, \ldots, g_n) \in \mathcal{G}_p^{\ n}$ and $x = (x_1, \ldots, x_n) \in \mathbb{Z}_p^{\ n}$, let

$$g^x := g_1^{x_1} \bullet \ldots \bullet g_n^{x_n}.$$

(Note that since the order of $\mathcal{G}_p$ is p, exponents only need to be defined modulo p.)

For $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n) \in \mathbb{Z}_p^{\ n}$, denote the inner product by

$$x * y := x_1 y_1 + \ldots + x_n y_n.$$

If m is a message, let $\underline{m}$ denote its extension to a vector

$$\underline{m} := (1, 1, m).$$

Names: We will call the signer Sibyl and the recipient Rick.

4 Overall system structure for one message of fixed length

To give a better idea of the scheme, we first assume that Sibyl will have to sign just one message of fixed length, and for a given recipient Rick. Note, however, that like with conventional signatures, every participant who knows the public keys can check the verification or disavowal of a signature, i.e. there is no a priori restriction on the participants later serving as a court. (Efficient extensions to many messages of arbitrary length and more recipients are shown in § 7.)

Remark: The difference between the recipient of a signature and someone who just checks the verification or disavowal of a signature (a difference that does not exist in conventional signatures) is that in this basic scheme, Rick has enough information to help Sibyl disavow an authentic signatures that he has received from her. However, this would only harm himself.

Key exchange:

0. All participants agree on a group $\mathcal{G}_p$ of prime order p and a parameter l. (l determines that the probability of successful cheating in the following zero-knowledge proofs should be at most 2^{-l}.)

1. The recipient Rick chooses a triple

$$g = (g_1, g_2, g_3)$$

of generators of $\mathcal{G}_p$, i.e. any elements of $\mathcal{G}_p{}^*$, with $g_1 \neq g_2$, randomly and publishes it.

The public values p, l, and g are parameters of the following algorithms, but will be omitted for simplicity.

2. The signer Sibyl checks that $g_1 \neq 1$ and $g_1 \neq g_2$. Then she chooses a secret key

$$SK = (x_1, x_2, x_3) \in \mathbb{Z}_p{}^3$$

randomly and computes and publishes the public key

$$PK = pub(SK) := g^{SK} = g_1{}^{x_1} \bullet g_2{}^{x_2} \bullet g_3{}^{x_3}.$$

Signing:
To sign a message $m \in \mathbb{Z}_p$, Sibyl forms the signature

$$sign(SK, m) := SK * \underline{m} = x_1 + x_2 + m \bullet x_3$$

Verification:
To accept a value $s \in \mathbb{Z}_p$ as a signature of m, Rick requires Sibyl to give a perfect zero-knowledge proof of knowledge (cf. [ToW1_87]) of

$$\text{I know } SK \in \mathbb{Z}_p{}^3\text{: } PK = pub(SK) \;\wedge\; sign(SK, m) = s. \tag{1}$$

(where PK, m, and s are common inputs).

Disavowal:
To disavow a value $sf \in \mathbb{Z}_p$ as a signature of m, the signer gives a perfect zero-knowledge proof of knowledge of

$$\text{I know } SK \in \mathbb{Z}_p{}^3\text{: } PK = pub(SK) \;\wedge\; sign(SK, m) \neq sf. \tag{2}$$

(Again, PK, m, and sf are common inputs.)

The description of efficient zero-knowledge proofs for these two purposes is postponed to § 6.

5 Security

5.1 Security of the signer

We must prove that with all the information a forger Felix can obtain from Sibyl, if he forges a signature, the probability that Sibyl can disavow it is exponentially close to 1. We first show this without the information obtained from verifications and disavowals. This means that Felix knows at most PK and one authentic signature s under a message m, and he wants to forge a signature under a different message m^*.

Lemma 1: $\forall$ *PK*, m, m^* with $m \neq m^*$, in the probability space defined by the random choice of the secret key *SK*:

If a forger Felix knows $PK = pub(SK)$ and $s = sign(SK, m)$, then for any forged signature *sf* for m^*, the probability that Sibyl can disavow *sf* is at least $1 - p^{-1}$.

Proof: Since the disavowal protocol is a proof of knowledge, it is, in particular, complete. This means that an incorrect signature can be disavowed with probability 1. Thus it suffices to prove for all $sf \in \mathbb{Z}_p$ that the conditional probability that *sf* is incorrect is $1 - p^{-1}$:

$$\Pr(sf \neq sign(SK, m^*) \mid PK = pub(SK) \text{ and } s = sign(SK, m)) = 1 - \frac{1}{p}.$$

Since the secret keys are uniformly distributed, this is equivalent to

$$\frac{|\{SK \mid sf = sign(SK, m^*) \wedge PK = pub(SK) \wedge s = sign(SK, m)\}|}{|\{SK \mid PK = pub(SK) \wedge s = sign(SK, m)\}|} = \frac{1}{p}. \tag{3}$$

Assume $g_2 = g_1{}^{\alpha}$, $g_3 = g_1{}^{\beta}$ and $PK = g_1{}^{\gamma}$. (Note that this representation is possible, since g_1 is a generator. Also note that we do not require any participant to compute these exponents, they just exist.)

Then $$PK = pub(SK) \Leftrightarrow g_1{}^{\gamma} = g_1{}^{x_1 + \alpha x_2 + \beta x_3} \Leftrightarrow \gamma = x_1 + \alpha x_2 + \beta x_3 \text{ in } \mathbb{Z}_p, \tag{4}$$

$$s = sign(SK, m) \Leftrightarrow s = x_1 + x_2 + m \bullet x_3, \tag{5}$$

and $$sf = sign(SK, m^*) \Leftrightarrow sf = x_1 + x_2 + m^* \bullet x_3. \tag{6}$$

The matrix of Equations (4), (5), and (6) can be transformed by row operations as

$$\begin{pmatrix} 1 & \alpha & \beta \\ 1 & 1 & m \\ 1 & 1 & m^* \end{pmatrix} \rightarrow \begin{pmatrix} 1 & \alpha & \beta \\ 0 & 1-\alpha & m-\beta \\ 0 & 0 & m^*-m \end{pmatrix}$$

We have $m \neq m^*$ and explicitly required $g_1 \neq g_2$, i.e. $\alpha \neq 1$. Thus the rank of this matrix is 3. Hence the equation system in the numerator of (3) has exactly one solution, whereas the system in the denominator has p solutions. This proves (3). □

Remark 1: Note that p must indeed be prime in Lemma 1. In the proof, this was needed for the notion of rank to be defined. More generally, otherwise, the equation corresponding to the last line of the matrix would sometimes have more than one solution. If there are d solutions, the probability that Sibyl can disavow *sf* is only $1 - d \bullet p^{-1}$.

Remark 2: Note that Sibyl's security depends on $g_1 \neq g_2$ only, not on the randomness of the generators. Thus it does not matter for her if Rick chooses them differently.

Also note that we have provided for an (adaptive) chosen message attack, since "$\forall$ m, m^*" means that Felix can choose both m and m^* in any order.

Remark 3: This proof can be seen as a special case of more general notions: The problem of preventing Rick or an outside forger Felix from guessing the correct signature for m^* from that for m is like the problem of authentication codes [GiMS_74]. If Felix is unrestricted, he is dealing with the family of functions $(sign(SK, \bullet))_{SK \in pub^{-1}(PK)}$. Therefore, a sufficient condition on security is that this family is strongly universal$_2$ [WeCa_81]. It can be shown that for each PK, our family is equal to the family H_1 from [CaWe_79].

Of course, Felix, at least if he cooperates with Rick, does not just see PK and s, but he may ask Sibyl to perform verifications of s and disavowals of other signatures. Hence we need the following lemma:

Lemma 2: No matter how the verifier behaves, Sibyl does not disclose any information about SK in the verification protocol; and if she performs the disavowal protocol for PK, m^*, and sf, she does not disclose any information about SK, except that $sign(SK, m^*) \neq sf$.

Sketch of proof: This is a consequence of the fact that verification and disavowal are perfect zero-knowledge. (This is not completely trivial, since perfect zero-knowledge is formulated with polynomially bounded verifiers and computational knowledge. One can also quite easily see it directly for the protocols in § 5.) □

If we combine these lemmas, we need one restriction: There are only p possible signatures for m^*, thus we cannot allow Felix to try them all. For example, we restrict him to $\sqrt{p}$ attempts. This does not contradict unconditional security, since it is a restriction not on Felix's computing abilities, but on the number of disavowals Sibyl or a court are willing to perform. In practice, for a realistic size of p, $\sqrt{p}$ disavowals are impossible anyway, since $\sqrt{p}$ is still exponential in the security parameter. Hence we obtain:

Theorem 1 (Sibyl's security): $\forall$ PK, m: Assume a forger Felix knows $PK = pub(SK)$ and $s = sign(SK, m)$ and can ask Sibyl to verify s arbitrarily often and disavow up to $\sqrt{p}$ adaptively chosen other signatures, i.e. pairs $(m^*, sf) \neq (m, s)$. Then the probability that Sibyl can disavow them all is at least $1 - \sqrt{p}^{-1}$.

Proof: From each disavowal, Felix can at most find out $sign(SK, m^*) \neq sf$ for one pair (m^*, sf). By the proof of Lemma 1, this excludes exactly one secret key. Thus no strategy gives him a better chance than $\sqrt{p}$ guesses for SK. The probability that he guesses right at least once is at most $\sqrt{p}\,/\,p = \sqrt{p}^{-1}$. □

The **invisibility** of the signatures, i.e. that they cannot be recognized without the help of the signer, follows from the fact that for someone who does not know SK, signatures look just like random elements of $\mathbb{Z}_p$, and that the verification and disavowal protocols are perfect zero-knowledge.

5.2 Security of the recipient

We first show that pub is cryptographically collision-free, i.e. Sibyl cannot find two secret keys fitting the same public key. A bit more generally, we prove the following lemma:

Lemma 3 (Collision-freeness of *n*-tuple exponentiation): Let $(\mathcal{G}_p)_p$ be a family of groups of prime orders p where the discrete log is hard, and n fixed. Then, for any probabilistic polynomial-time algorithm $\mathcal{A}_n$, any polynomial Q, and sufficiently large

p: The probability that $\mathcal{A}_n$, on input a random n-tuple $G_n = (g_1, ..., g_n)$ of generators of $\mathcal{G}_p$, finds a G_n-collision, i.e.

$$X_n \neq X_n' \in \mathbb{Z}_p{}^n \text{ with } G_n{}^{X_n} = G_n{}^{X_n'},$$

is smaller than $1/Q(\log(p))$.

Proof: For any tuple G_n of generators, let

$$Coll(G_n)$$

be the set of G_n-collisions. The proof is by induction on n. The case $n = 1$ need not be considered, because collisions do not exist at all for $n = 1$.

***n* = 2:**

Assume that $\mathcal{A}_2$ is a polynomial-time algorithm which finds collisions for pair exponentiation with significant probability (i.e. at least $1/Q(\log(p))$ for some Q and infinitely many p). From $\mathcal{A}_2$, we construct an algorithm $\mathcal{A}_1$ which breaks the discrete log.

$\mathcal{A}_1$: Input: g, $h \neq 1$ (where y with $h = g^y$ is to be found).
Choose e_1, e_2 randomly from $\mathbb{Z}_p{}^*$; let $g_1 := g^{e_1}$; $g_2 := h^{e_2}$;
(i.e. blind g, h to form random generators)
$C_2 \leftarrow \mathcal{A}_2(g_1, g_2)$;
(i.e., run $\mathcal{A}_2$ with the inputs g_1, g_2 and call the result C_2)
If C_2 is a collision $(X_2, X_2') = ((x_1, x_2), (x'_1, x'_2))$, then output
$y := e_1\,(x'_1–x_1)\,e_2{}^{-1}\,(x'_2–x_2)^{-1}$. (Otherwise anything.)

If C_2 is a collision, this means $g_1{}^{x_1} \bullet g_2{}^{x_2} = g_1{}^{x'_1} \bullet g_2{}^{x'_2}$. Also $x'_2 \neq x_2$, since $x'_2 = x_2$ would imply $g_1{}^{x_1} = g_1{}^{x'_1}$ and therefore $x'_1 = x_1$, in contradiction to $X_2 \neq X_2'$. Thus $e_1 \bullet (x'_1–x_1) \bullet e_2{}^{-1} \bullet (x'_2–x_2)^{-1}$ exists.

The first equation implies

$$g^{e_1x_1} \bullet h^{e_2x_2} = g^{e_1x'_1} \bullet h^{e_2x'_2} \Rightarrow g^{e_1x_1 - e_1x'_1} = h^{e_2x'_2 - e_2x_2}$$
$$\Rightarrow h = g^{e_1(x_1-x'_1)\,e_2{}^{-1}(x'_2-x_2)^{-1}} = g^y.$$

Thus, if $\mathcal{A}_2$ has found a collision, $\mathcal{A}_1$ has found the discrete log. Since the inputs to $\mathcal{A}_2$ within $\mathcal{A}_1$ are completely random generators, the probability that $\mathcal{A}_1$ succeeds on the given input is just as high as the global success probability of $\mathcal{A}_2$. Obviously, $\mathcal{A}_1$ is polynomial-time, too.

Induction step; *n* > 2:

Assume that $\mathcal{A}_n$ is an algorithm which finds collisions for n-tuples with significant probability (i.e. at least $1/Q(\log(p))$ for some Q and infinitely many p). From $\mathcal{A}_n$, we construct an algorithm $\mathcal{A}_{n-1}$ which finds collisions for $(n–1)$-tuples with almost the same probability. It

- first transforms the given $(n–1)$-tuple G_{n-1} of generators into a completely random n-tuple G_n of generators (some sort of blinding),
- then uses $\mathcal{A}_n$ as subprogram with input G_n to obtain a G_n-collision C_n with significant probability,
- and finally transforms C_n back into something called C_{n-1}, which should be a G_{n-1}-collision.

a) We first define the two (deterministic) **transformation functions**. The generator-extension ***ext***, the exponent-reduction ***red_exp***, and the collision-reduction ***red_coll*** are defined as: For any tuples

$E = (e_1, ..., e_{n-1}) \in (\mathbb{Z}_p^{\ *})^{n-1}$,
$R = (r_1, ..., r_{n-1}) \in \mathbb{Z}_p^{\ n-1}$,
$G_{n-1} = (g_1, ..., g_{n-1}) \in \mathcal{G}_p^{\ n-1}$, and
$X_n = (x_1, ..., x_n) \in \mathbb{Z}_p^{\ n}$,
$C_n = (X_n, X_n') \in (\mathbb{Z}_p^{\ n})^2$,

let

$$ext(G_{n-1}, E, R) := (g_1^{\ e_1}, \ldots, g_{n-1}^{\ e_{n-1}}, g_1^{\ r_1} \bullet \ldots \bullet g_{n-1}^{\ r_{n-1}}),$$
$$red\_exp(X_n, E, R) := (e_1x_1 + r_1x_n, \ \ldots, e_{n-1}x_{n-1} + r_{n-1}x_n),$$
$$red\_coll(C_n, E, R) := (red\_exp(X_n, E, R), red\_exp(X_n', E, R)).$$

b) Thus our **algorithm** is:

$\mathcal{A}_{n-1}$: On input G_{n-1}, where G_{n-1} is an $(n-1)$-tuple $G_{n-1} = (g_1, ..., g_{n-1})$:

1. Choose $e_1, ..., e_{n-1}$ randomly from $\mathbb{Z}_p^{\ *}$ and $r_1, ..., r_{n-1}$ randomly from $\mathbb{Z}_p$, and let
$$E := (e_1, ..., e_{n-1}) \text{ and } R := (r_1, ..., r_{n-1}).$$
2. Define an n-tuple
$$G_n := (g_1', ..., g_n') := ext(G_{n-1}, E, R).$$
If the last component of G_n is not a generator, repeat the choice of r_{n-1} until it is.
3. Run $\mathcal{A}_n$ on this n-tuple and call the result C_n, i.e.
$$C_n \leftarrow \mathcal{A}_n(G_n);$$
If C_n is a collision $C_n = (X_n, X_n')$, output
$$C_{n-1} := (X_{n-1}, X_{n-1}') := red\_coll(C_n, E, R).$$

c) $\mathcal{A}_{n-1}$ is **successful** iff its output C_{n-1} is a G_{n-1}-collision, thus let (cf. § 3.1)

$$P_{succ}(\mathcal{A}_{n-1}, G_{n-1}) := \mathrm{P}_{\mathcal{A}_{n-1}(G_{n-1})}(Coll(G_{n-1})) = \sum_{C_{n-1} \in Coll(G_{n-1})} \mathrm{P}_{\mathcal{A}_{n-1}(G_{n-1})}(C_{n-1}). \quad (7)$$

To compute the probability of outputs of $\mathcal{A}_{n-1}$, we consider it as the composition of its three steps 1. through 3. (Step 2. is deterministic, but that is just a special case of a probabilistic algorithm.) For this, we need a name for the algorithm in Step 1, say *rand*. Thus Step 1. is $(E, R) \leftarrow rand()$. Now literal application of the last formula of § 3.1 yields

$$\mathrm{P}_{\mathcal{A}_{n-1}(G_{n-1})}(C_{n-1}) =$$
$$= \sum_{(E,R)} \mathrm{P}_{rand()}((E,R)) \bullet \sum_{G_n} \mathrm{P}_{ext(G_{n-1},E,R)}(G_n) \bullet \sum_{C_n} \mathrm{P}_{\mathcal{A}_n(G_n)}(C_n) \bullet 1_{red\_coll(C_n,E,R)=C_{n-1}}.$$

In this formula, we can evaluate the first two probabilities, since *rand* is an easy algorithm and *ext* is deterministic: All R's and E's are equally probable. There are

- $p-1$ choices for each e_i,
- p choices for each r_i with $i \le n-2$, and
- $p-1$ choices for r_{n-1} (the p-th value of r_{n-1} would yield g_n'=1).

Thus

$$\mathrm{P}_{\mathcal{A}_{n-1}(G_{n-1})}(C_{n-1}) =$$

$$= \sum_{(E,R)} \frac{1}{(p-1)^n \bullet p^{n-2}} \bullet \sum_{G_n} 1_{ext(G_{n-1},E,R)=G_n} \bullet \sum_{C_n} \mathrm{P}_{\mathcal{A}_n(G_n)}(C_n) \bullet 1_{red\_coll(C_n,E,R)=C_{n-1}}. \tag{8}$$

d) We now substitute (8) into (7), divide the constant fraction out and reorder the sums (with the aim that we want to use the success probability of $\mathcal{A}_n$). We abbreviate

$$\boldsymbol{P^*} := P_{succ}(\mathcal{A}_{n-1}, G_{n-1}) \bullet (p-1)^n \bullet p^{n-2}.$$

Thus

$$P^* = \sum_{G_n} \sum_{C_n} \mathrm{P}_{\mathcal{A}_n(G_n)}(C_n) \bullet \sum_{(E,R)} 1_{ext(G_{n-1},E,R)=G_n} \bullet \sum_{C_{n-1} \in Coll(G_{n-1})} 1_{red\_coll(C_n,E,R)=C_{n-1}}.$$

We further transform this by restricting ourselves to G_n-collisions in the second sum (i.e. we only consider the case where $\mathcal{A}_n$ is successful), and by evaluating the last sum (which has at most one non-zero term):

$$P^* \geq \sum_{G_n} \sum_{C_n \in Coll(G_n)} \mathrm{P}_{\mathcal{A}_n(G_n)}(C_n) \bullet \sum_{(E,R)} 1_{ext(G_{n-1},E,R)=G_n} \bullet 1_{red\_coll(C_n,E,R) \in Coll(G_{n-1})}. \tag{9}$$

The term which is now the last sum is just

$$| \{(E, R) \mid ext(G_{n-1}, E, R) = G_n \wedge red\_coll(C_n, E, R) \in Coll(G_{n-1})\} |. \tag{10}$$

e) We will now compute this **cardinality** (for arbitrary, but fixed, G_{n-1}, G_n, and $C_n \in Coll(G_n)$).

e1) The first $n-1$ components of the first equation of (10) are of the form $g_i' = g_i^{e_i}$. They have a unique solution E. The last component of the first equation of (10) is of the form $g_n' = g_1^{r_1} \bullet \ldots \bullet g_{n-1}^{r_{n-1}}$, which leaves p^{n-2} choices for R.

e2) We prove that for all but one of them, the second equation is fulfilled, too. This equation means

$$G_{n-1}^{red\_exp(X_n, E, R)} = G_{n-1}^{red\_exp(X_n', E, R)}$$
$$\wedge \quad red\_exp(X_n, E, R) \neq red\_exp(X_n', E, R). \tag{11}$$

e2α) Let $X_n = (x_1, \ldots, x_n)$ and $X'_n = (x'_1, \ldots, x'_n)$.

By definition, $red\_exp(X_n, E, R) = (e_1 x_1 + r_1 x_n, \ldots, e_{n-1} x_{n-1} + r_{n-1} x_n)$, thus

$$G_{n-1}^{red\_exp(X_n, E, R)} = g_1^{e_1 x_1 + r_1 x_n} \bullet \ldots \bullet g_{n-1}^{e_{n-1} x_{n-1} + r_{n-1} x_n}.$$

We will use that C_n is a G_n-collision, i.e. $G_n^{X_n} = G_n^{X_n'}$, where

$$G_n^{X_n} = g_1'^{\,x_1} \bullet \ldots \bullet g_n'^{\,x_n} = (g_1^{e_1})^{x_1} \bullet \ldots \bullet (g_{n-1}^{e_{n-1}})^{x_{n-1}} \bullet (g_1^{r_1} \bullet \ldots \bullet g_{n-1}^{r_{n-1}})^{x_n}$$
$$= G_{n-1}^{red\_exp(X_n, E, R)},$$

and, similarly

$$G_n^{X_n'} = G_{n-1}^{red\_exp(X_n', E, R)}.$$

Thus the first equation of (11) is always fulfilled. (This is, of course, how *ext* and *red_coll* were chosen.)

e2β) Thus we have left to show in Part e2) that there cannot be more than one R with $red\_exp(X_n, E, R) = red\_exp(X_n', E, R)$.

Assume, to the contrary, that there were two tuples $R^{(1)} \neq R^{(2)}$ which fulfilled this equation; let $R^{(1)} = (r^{(1)}{}_1, ..., r^{(1)}{}_{n-1})$, $R^{(2)} = (r^{(2)}{}_1, ..., r^{(2)}{}_{n-1})$. The equation for the i-th components in the equation $red\_exp(X_n, E, R^{(1)}) = red\_exp(X_n', E, R^{(1)})$ is

$$e_i x_i + r^{(1)}{}_i\, x_n = e_i x'_i + r^{(1)}{}_i\, x'_n. \qquad (i1)$$

Similarly, for $R^{(2)}$,

$$e_i x_i + r^{(2)}{}_i\, x_n = e_i x'_i + r^{(2)}{}_i\, x'_n. \qquad (i2)$$

Equations $(i1)$ and $(i2)$ imply

$$(r^{(1)}{}_i - r^{(2)}{}_i) \bullet x_n \;=\; (r^{(1)}{}_i - r^{(2)}{}_i) \bullet x'_n. \qquad (i3)$$

Because of $R^{(1)} \neq R^{(2)}$, at least one of the equations $(i3)$ implies $x_n = x'_n$. Now Equation $(i1)$ becomes

$$e_i x_i \;=\; e_i x'_i \;\Rightarrow\; x_i \;=\; x'_i \;\text{ (for } i = 1, \ldots, n\text{–}1).$$

Thus $(x_1, ..., x_n) = (x'_1, ..., x'_n)$, in contradiction to the fact that C_n is a collision.

Putting e1) and e2) together, we have proved

$$|\{(E,R) \mid ext(G_{n-1}, E, R) = G_n \wedge red\_coll(C_n, E, R) \in Coll(G_{n-1})\}| \;\geq\; p^{n-2} - 1.$$

f) Substituting this and (10) into (9), we obtain

$$P^* \;\geq\; \sum_{G_n} \sum_{C_n \in Coll(G_n)} \mathrm{P}_{\mathcal{A}_n(G_n)}(C_n) \bullet (p^{n-2} - 1) \;=\; (p^{n-2} - 1) \bullet \sum_{G_n} \sum_{C_n \in Coll(G_n)} \mathrm{P}_{\mathcal{A}_n(G_n)}(C_n). \qquad (12)$$

Similarly to (7), the success probability of $\mathcal{A}_n$ on input G_n is defined as

$$P_{succ}(\mathcal{A}_n, G_n) := \mathrm{P}_{\mathcal{A}_n(G_n)}(Coll(G_n)) = \sum_{C_n \in Coll(G_n)} \mathrm{P}_{\mathcal{A}_n(G_n)}(C_n), \qquad (13)$$

and since there are $(p\text{–}1)^n$ n-tuples of generators, the overall success probability of $\mathcal{A}_n$ is

$$P_{succ}(\mathcal{A}_n) := \frac{1}{(p-1)^n} \bullet \sum_{G_n} P_{succ}(\mathcal{A}_n, G_n). \qquad (14)$$

Substituting (13) and (14) in (12) yields

$$P^* \;\geq\; (p^{n-2} - 1) \bullet \sum_{G_n} P_{succ}(\mathcal{A}_n, G_n) \;=\; (p^{n-2} - 1) \bullet (p-1)^n \bullet P_{succ}(\mathcal{A}_n).$$

Remembering the definition of P^* (in d), we see that this means

$$P_{succ}(\mathcal{A}_{n-1}, G_{n-1}) \;\geq\; \frac{p^{n-2} - 1}{p^{n-2}} \bullet P_{succ}(\mathcal{A}_n).$$

Thus, for each G_{n-1}, $P_{succ}(\mathcal{A}_{n-1}, G_{n-1})$ is almost $P_{succ}(\mathcal{A}_n)$, i.e., $\mathcal{A}_{n-1}$ finds G_{n-1}-collisions with this probability. Consequently, the overall success probability of $\mathcal{A}_{n-1}$ is also almost $P_{succ}(\mathcal{A}_n)$. (More precisely, it is at least $1/2Q(\log(p))$ for infinitely many p.) This contradicts the presupposition of this induction step that finding collisions for (n–1)-tuple exponentiation is hard. Hence the assumed algorithm $\mathcal{A}_n$ cannot exist. This finishes the proof of the induction step and thus of the theorem. □

Remark: Note that p must indeed be prime in Lemma 3: In the proof, this is needed where inverses of exponents from $\mathbb{Z}_p^{\,*}$, such as e_2^{-1}, are used.

As an example that Lemma 3 is indeed false for general p, assume $p = 2q$. Then $\mathcal{G}_p$ has exactly one element w of order 2. (Note that $\mathcal{G}_p$ would be assumed to be cyclic, since we talk about generators.) For each generator g, we have $g^q = w$. Already for two generators g_1, g_2, this yields $g_1{}^q\, g_2{}^0 = w = g_1{}^0\, g_2{}^q$. Hence one can easily find the collision $((q, 0), (0, q))$.

More generally, if $p = f{\bullet}q$ for any small factor f, then $g := g_1{}^q$ and $h := g_2{}^q$ are elements of the small subgroup of order f. Hence one can easily find the discrete log e of h with respect to g. Then one obtains $g_1{}^{q \bullet e}\, g_2{}^0 = g_1{}^0\, g_2{}^q$.

Remark: Lemma 3 is related to the "relaxed discrete log" problem mentioned in [ChEG_88]. There, it is stated (without proof) that solving equations $G_n{}^{X_n} = h$, where $G_n = (g_1, ..., g_n)$ and h are chosen randomly, and n may grow polynomially in $\log(p)$, is hard, also if p is not prime. For p prime and n fixed, this statement follows from Lemma 3. An implication in the other direction is not known. (One might say that this statement means that n-tuple exponentiation is a one-way function, whereas ours meant that the function is collision-free.)

Theorem 2 (Rick's security): If the discrete log in the chosen family of groups is hard, it is infeasible for Sibyl to prove an s to be a valid signature for a message m and to disavow it later.

Proof: The soundness of the interactive proofs implies that if Sibyl can prove s to be a valid signature for m and later disavow it, she can compute secret keys SK and SK^* with

$$PK = pub(SK) \;\wedge\; sign(SK, m) = s \;\wedge\; PK = pub(SK^*) \;\wedge\; sign(SK^*, m) \neq s.$$

Since $sign$ is deterministic, this implies $SK \neq SK^*$, i.e. Sibyl has found a collision of the function pub. This contradicts Lemma 3. □

6 Efficient verification and disavowal

It remains to present efficient perfect zero-knowledge proofs for verification and disavowal.

6.1 Verification

We use the following Protocol 1:

Repeat l times (see § 4, Point 0):

Sibyl (Prover)		Rick (Verifier)
Choose $R = (r_1, r_2, r_3) \in \mathbb{Z}_p{}^3$ randomly. Compute $SK' = SK + R$, $PK' = pub(SK')$, and $s' = sign(SK', m)$.		
	$\xrightarrow{PK',\ s'}$	Choose $b \in \{0,1\}$ randomly.
	$\xleftarrow{\quad b \quad}$	
(If b=0, open the blinding R. If b=1, prove the knowledge	$\xrightarrow{R \text{ or } SK'}$ if $b = 0$ or 1, resp.	

for the blinded problem instance.) If b=0, check that $PK' = PK \cdot g^R$
$\wedge\ s' = s + R * \underline{m}$.
If b=1, check that $PK' = pub(SK')$
$\wedge\ s' = sign(SK', m)$.

Protocol 1 Verification of a signature *s*

We will prove that Protocol 1 is a perfect zero-knowledge interactive proof of knowledge for the relation defined by Formula (1). Briefly, this holds because Protocol 1 is a special case of the proof system for random self-reducible relations of [ToW1_87, Theorem 4]. The self-reduction in our case is exactly the addition of a random R to the secret key SK. (In general, blinding and random self-reduction are about the same thing.)

For a more detailed proof, we will generalize Lemma 4 a bit, so that the structure can be seen more clearly.

Lemma 4: Let (G_N), (H_N) be families of groups. Assume that random elements in G_N can be chosen efficiently (i.e. polynomially in N), and that the group operations can be computed efficiently. Let ϕ_N: $G_N \rightarrow H_N$ be efficiently computable group homomorphisms.

Then the family of converses of the graphs of ϕ_N (i.e. $\{(h, g) \mid h = \phi_N(g)\}$ is random self-reducible (in the sense of [ToW1_87]).

Proof: (In the notation of [ToW1_87], the parameter N of the relation is the same as the N here, the problem instance x is our h, and the solution y is our g.)

The idea is that a solution g is blinded as $blind(g, r) := g \bullet r$ with a random $r \in G_N$. Thus the algorithm A, which is to transform the problem instances correspondingly, is defined by

$$A(N, h, r) := h \bullet \phi_N(r).$$

Obviously, A is efficient.

Remark: More precisely, in [ToW1_87], r ist just a random bit string. Hence we would need the assumed algorithm *rand* which chooses random elements in G_N, and define $A(N, h, r) := h \bullet \phi_N(rand(N, r))$.

We need the set $Solv_N$ of all solvable problem instances, which is called $dom\ R'_N$ in [ToW1_87]. Here, it is the image $im(\phi_N)$. We must prove the requirements R1, R2, R3, T1, and T2 from [ToW1_87].

R1 **Uniformity:** (This property means that the blinded problem instances are randomly distributed over all solvable problem instances.) First, each blinded problem instance is solvable again, since $A(N, h, r) = h \bullet \phi_N(r) \in im(\phi_N)$ if $h \in im(\phi_N)$. Secondly, for each $h^* \in Solv_N = im(\phi_N)$, we have

$$A(N, h, r) = h^* \Leftrightarrow \phi_N(r) = h^{-1} \bullet h^*,$$

with $h^{-1} \bullet h^* \in im(\phi_N)$. The set of solutions r to this equation is a coset of $ker(\phi_N)$. Thus its cardinality is independent of h^*.

R2 **Blinded solution $\Rightarrow$ original solution:** Given are N, h, r, and a solution to the blinded problem, i.e. g' with $\phi_N(g') = h' := A(N, h, r)$.

Choose (more precisely: define the required transformation algorithm so that it chooses) $g := g' \bullet r^{-1} = blind(g', r^{-1})$. Then

$$\phi_N(g) = \phi_N(g') \bullet \phi_N(r)^{-1} = h' \bullet \phi_N(r)^{-1} = (h \bullet \phi_N(r)) \bullet \phi_N(r)^{-1} = h.$$

R3 Original solution $\Rightarrow$ blinded solution uniformly: Given are N, h, r, and $g \in \phi_N^{-1}(h)$. Choose (see R2) $g' := blind(g, r) = g \bullet r$.

First we show that this is a solution to the blinded problem $h' := A(N, h, r)$:

$$\phi_N(g') = \phi_N(g) \bullet \phi_N(r) = h \bullet \phi_N(r) = h'.$$

Secondly, we need that the values $blind(g, r)$ for all r's with $A(N, h, r) = h'$ are uniformly distributed over all solutions to h'. For this, let such a solution $g^* \in \phi_N^{-1}(h')$ be given.

Since r is uniformly distributed, it suffices to show that the number of r's with $blind(g, r) = g^* \wedge A(N, h, r) = h'$, i.e. $g \bullet r = g^* \wedge h \bullet \phi_N(r) = h'$ is independent of g^*. There is at most one, namely $r^* = g^{-1} \bullet g^*$. So we only need to check the second equation for it:

$$h \bullet \phi_N(r^*) = h \bullet \phi_N(g)^{-1} \bullet \phi_N(g^*) = h \bullet h^{-1} \bullet h' = h'.$$

The transformation algorithms in R2 and R3 are obviously polynomial.

T1 Testability of solutions: We must show that, given N, g, and h, it can be efficiently checked if $\phi_N(g) = h$. This is obvious, since ϕ_N is efficiently computable.

T2 Accessability: This property means that random problems with known solution can be generated efficiently. More precisely, the problem must be uniformly distributed among all solvable problems, and the solution uniformly distributed among the solutions to this problem.

We use an algorithm D which, given N, chooses $g \in G_N$ randomly and computes $h := \phi_N(g)$. According to the assumptions, D is efficient. The uniformity of h follows from the fact that each $h \in im(\phi_N)$ has the same number of preimages (see R1). The uniformity of g is trivial. □

Corollary (of Lemma 4 and [ToW1_87, Theorem 4]): On the assumptions of Lemma 4, the following Protocol 2 is a perfect zero-knowledge interactive proof of knowledge for the the family of converses of the graphs of ϕ_N. In our more intuitive notation, this means a proof of knowledge of

I know g: $\phi_N(g) = h$.

Repeat l times:

Sibyl (Prover)		Rick (Verifier)
Choose $r \in G_N$ randomly. Compute $g' := g \bullet r$ and $h' = \phi_N(g')$		
	$\xrightarrow{h'}$	Choose $b \in \{0,1\}$ randomly.
	$\xleftarrow{b}$	
(If b=0, open the blinding r. If b=1, prove the knowledge for the blinded problem instance.)	$\xrightarrow{r \text{ or } g'}$ if b = 0 or 1, resp.	
		If b=0, check that $h' = h \bullet \phi_N(r)$ If b=1, check that $h' = \phi_N(g')$.

Protocol 2 General proof of the knowledge of a preimage of a homomorphism

Lemma 5: Protocol 1 is a perfect zero-knowledge interactive proof of knowledge for the relation defined by Formula (1).

Proof: We show that Protocol 1 is a special case of Protocol 2. We have

pub_p: $\mathbb{Z}_p{}^3 \to \mathcal{G}_p$ with $pub_p((x, y, z)) = g_1{}^{x_1} \bullet g_2{}^{x_2} \bullet g_3{}^{x_3}$, and

$sign_p(\bullet\ , m)$: $\mathbb{Z}_p{}^3 \to \mathbb{Z}_p$ with $sign_p((x, y, z), m) = x + y + m \bullet z$.

$(\mathbb{Z}_p{}^3, +)$ is, of course, a group, and it is easy to see that pub_p and $sign_p(\bullet\ , m)$ are homomorphisms. Hence

$$\phi_{p,m}: \mathbb{Z}_p{}^3 \to \mathcal{G}_p \times \mathbb{Z}_p$$

with $$\phi_{p,m}(SK) = (pub_p(SK), sign_p(\bullet\ , m))$$

is a homomorphism, too. (This is the universal mapping property of the cartesian product $\mathcal{G}_p \times \mathbb{Z}_p$). Thus we want to apply the corollary with (p, m) as the parameter N, $G_N = (\mathbb{Z}_p{}^3, +)$ and $H_N = (\mathcal{G}_p, \bullet) \times (\mathbb{Z}_p, +)$.

The operations in $\mathcal{G}_p$ were assumed to be efficient (§ 3.2); the rest of the assumptions of Lemma 4 is obviously fulfilled. Thus the corollary can be applied. The resulting special case of Protocol 2 is exactly Protocol 1. □

6.2 Disavowal

6.2.1 Idea for the zero-knowledge protocols

Our zero-knowledge disavowal protocols have a structure similar to the disavowal protocol in [Chau1_90]. They are in l' rounds, with Rick allowed to choose one of three challenges in each round. l' is chosen so that $(2/3)^{l'} \le 2^{-l}$. We now consider one round.

We use the same blinding idea as for verification: Sibyl chooses R randomly and computes blinded values $SK' = SK + R$, $PK' = pub(SK')$, $sf' = sf + R * \underline{m}$, and $s' = sign(SK', m)$ (the blinded correct signature for the same message).

Remark: Note why disavowal is not so easy as verification: Assume that, similar to Protocol 1, Sibyl would send PK' and sf' to Rick, and Rick might ask her to show either R or SK'. (Whereupon Rick would check $PK' = PK \bullet g^R \wedge sf' = sf + R * \underline{m}$ or $PK' = pub(SK') \wedge sf' \ne sign(SK', m)$, resp.)

This protocol is not zero-knowledge: Suppose Rick falsely claims that Sibyl has signed m, and in the disavowal protocol, he asks her to show SK', i.e. $b = 1$.

Then Rick has the following equations (and some more), where only SK and R are unknown:

$$SK' = SK + R \tag{15}$$
$$sf' = sf + R * \underline{m} \tag{16}$$

Equation (15) implies

$$SK' * \underline{m} = SK * \underline{m} + R * \underline{m}$$

This and (16) yield

$$sign(SK, m) = SK * \underline{m} = SK' * \underline{m} + sf - sf'.$$

Hence Rick has found out the real signature under the message m for which he wanted to forge a signature. This is knowledge he will certainly appreciate.

To avoid the problem from the remark, Sibyl must not show both SK' and sf'. This is achieved in the following way:

She prepares commitments [BrCC_88] on PK', s', and sf' and sends only those to Rick in the first step (cf. Fig. 1; commitments are represented by enclosures). The commitments must be unconditionally hiding.

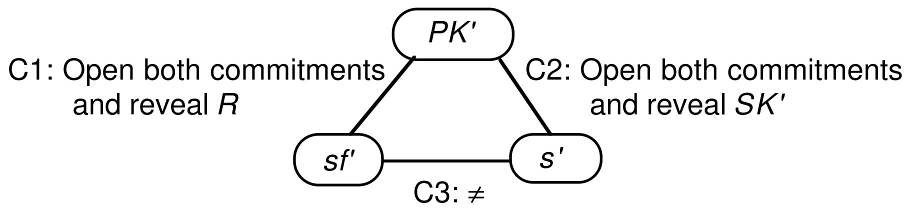

Fig. 1 Idea of the disavowal protocols

Ideally, Rick can choose from the following three challenges:

C1 Sibyl must open the two left commitments and reveal R. (Hence she proves that she has blinded correctly.)

C2 Sibyl must open the two right commitments and reveal SK'. (Hence she proves that s' is the correct signature under m in the blinded version.)

C3 Sibyl must prove inequality of the values in the lower two commitments, without opening the commitments. (Hence she proves, in the blinded version, that the correct signature s' is different from sf', which is to be disavowed.)

The problem is how to show inequality of the values in commitments without revealing any other information about them.

6.2.2 An easy, but not very efficient zero-knowledge disavowal

The protocol idea from § 6.2.1 can be realized with bit commitments, similar to [Chau1_90]. Since there is a better protocol in the following section, we only sketch this one:

The lower two commitments are made bitwise, and the bits are permuted with the same random permutation π. There is an additional commitment on π. (The upper commitment and that on π need not be bitwise, but must also be unconditionally hiding.) In Cases C1 and C2, the commitment on π is opened, too, so that the correct order of the bits can be restored. In Case C3, the inequality of just one pair of bits (in the same position) is shown. There are unconditionally hiding bit commitments which allow this without showing which bit is 1 and which is 0 [BrCC_88]. Thus in more detail, the commitment structure looks like in Fig. 2:

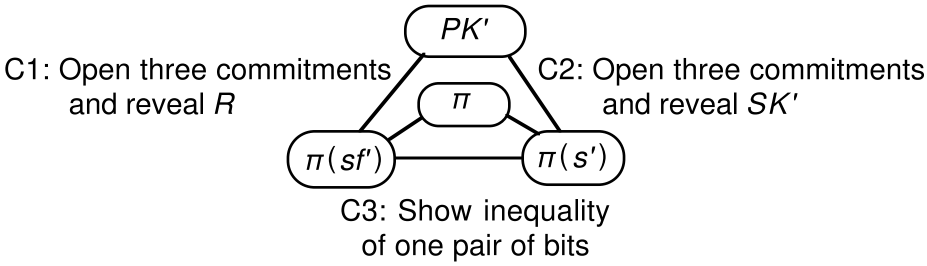

Fig. 2 Disavowal protocol with bit commitments

Remark: The structure of this protocol is not exactly the same as that in [Chau1_90] (since there, messages can be blinded, too).

But the protocol of Fig. 2 can be generalized so that it can be used for that case, too: We only need that *pub* and *sign*(•, *m*) are group homomorphisms, and write *pub*(*R*) instead of g^R, and *sign*(*R*, *m*) instead of $R{*}\underline{m}$.

Even more generally, given two homomorphisms $\phi: G \to H$ and $\psi: G \to I$, this is a perfect ZKP of knowledge of the formula ($h \in H$ and $i \in I$ are given)

$$\exists g \in G: h = \phi(g) \ \wedge \ \psi(g) \neq i.$$

(Of course, more formally this would look like Lemma 4.)

However, the bit commitments make this protocol far less efficient than the verification protocol, since they make the messages much longer. Although this might be tolerated in practice, since disavowals are rare (at least if one only performs them in real courts), we present a new second protocol, which is almost as efficient as the verification, using commitments on complete numbers instead of bits.

6.2.3 Efficient zero-knowledge disavowal

Commitment schemes are usually defined for single bits only. However, commitments can also be made on complete numbers in one piece. Unconditionally hiding commitments can be achieved by hiding functions [Pfit1_89]. Here we present a scheme based on the discrete log.

Lemma 6: If $\mathcal{G}_p$, g_1, and g_2 are chosen as in § 4, Sibyl can make unconditionally hiding commitments on elements of $\mathbb{Z}_p$ by using

$$a = h(\alpha, x) := g_1{}^{\alpha} \bullet g_2{}^{x}$$

with random x as a commitment on the first parameter α. It is opened by showing α and x.

Commitments can be added and subtracted, i.e. if Sibyl has made commitments b on β and c on γ, she can open $b{\bullet}c$ and b/c as commitments on $\beta{+}\gamma$ and $\beta{-}\gamma$, resp.

The commitments on a fixed value α are uniformly distributed in $\mathcal{G}_p$.

Proof: The content is unconditionally hidden in a commitment, since for all $a \in \mathcal{G}_p$, $\alpha \in \mathbb{Z}_p$, there is exactly one x with $h(\alpha, x) = a$. This also proves the last sentence.

Lemma 3 implies that finding a commitment that she can open in two different ways is as hard for Sibyl as the discrete log.

Finally, h is a homomorphism, i.e.

$$h(\beta, x) \bullet h(\gamma, y) = g_1{}^{\beta} \bullet g_2{}^{x} \bullet g_1{}^{\gamma} \bullet g_2{}^{y} = g_1{}^{\beta+\gamma} \bullet g_2{}^{x+y}$$
$$= h(\beta+\gamma, x+y) = h((\beta, x)+(\gamma, y)),$$

and similarly, $h(\beta, x) \,/\, h(\gamma, y) = h(\beta-\gamma, x-y) = h((\beta, x) - (\gamma, y)).$

Hence Sibyl can open $b \bullet c$ and b/c by showing $(\beta+\gamma, x+y)$ and $(\beta-\gamma, x-y)$, resp. □

In the disavowal protocol, Sibyl must prove the inequality of the contents of two commitments, without giving any additional information about those contents.

The basic idea of the following scheme is this: Since contents of commitments can be subtracted, it suffices to show that the value ε in a commitment $e = h(\varepsilon, z)$ is not zero. To make all nonzero ε's look alike, we use exponential blinding: For $r \in \mathbb{Z}$,

$$e^r = h(\varepsilon, z)^r = h(r\varepsilon, rz).$$

If r is chosen from $\{1, \ldots, p-1\}$ randomly, the product $r\varepsilon$ is uniformly distributed in $\mathbb{Z}_p{}^*$ for $\varepsilon \neq 0$, and always 0 for $\varepsilon = 0$. This suggests that Sibyl should either open e^r, or show that it has been formed correctly. However, she may not show e and open e^r, because e and e^r determine r and thus ε as $r\varepsilon / \varepsilon$. Hence an additional level of commitments is introduced.

Thus for the actual disavowal, Sibyl prepares a commitment structure as shown in Fig. 3.

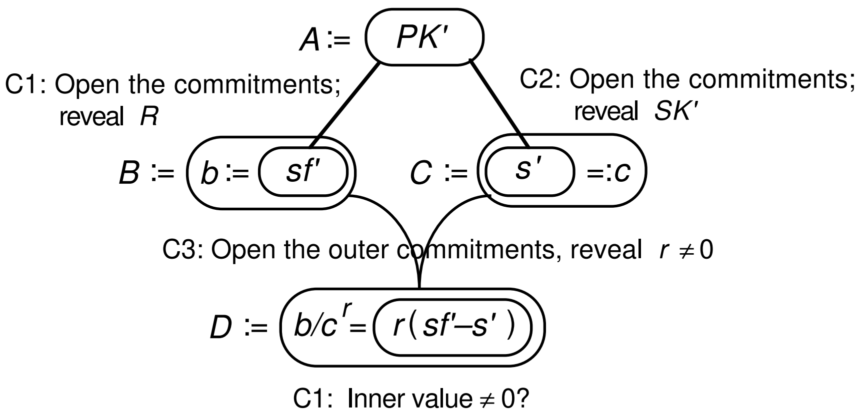

Fig. 3 Structure ot the efficient disavowal protocol

The values in the inner commitments, i.e. sf', s', and $r(sf'-s')$, are from $\mathbb{Z}_p$. Hence the same group $\mathcal{G}_p$ can be used for the inner commitments as for the signature scheme itself.

For the outer commitments, we need an efficient embedding ι: $\mathcal{G}_p \to \mathbb{Z}_{p'}$ for a possibly larger prime p'. We show that we can even use $p' = p$ in important cases. (Then ι is a bijection automatically. The discrete log is, of course, always a bijection $\mathcal{G}_p \to \mathbb{Z}_p$, too, but not efficiently computable.)

Lemma 7: For the first two choices of $\mathcal{G}_p$ in § 3.2, there are efficiently computable embeddings ι: $\mathcal{G}_p \to \mathbb{Z}_p$.

Proof: For the second choice of $\mathcal{G}_p$ in § 3.2, i.e. $\mathcal{G}_p = \mathbb{Z}_q{}^*/\{\pm 1\}$ where $q = 2p+1$ and q is also prime, $\mathcal{G}_p$ was represented by $\{1, \ldots, p\}$. Hence ι only needs to change p into 0.

For the first choice of $\mathcal{G}_p$ in § 3.2, i.e. $\mathcal{G}_p$ = GF(2^n)* where $p = 2^n–1$ is a Mersenne-prime, $\mathcal{G}_p$ is represented by the vectors from $\{0,1\}^n$ except for the zero vector. Thus ι only needs to interpret the vectors as binary representations of numbers and to replace 2^n by 0. □

(The third choice, being equivalent to the second one, is not so important. With general elliptic curves, the representation of a group element has two components of about the length of p, thus one might make two different commitments on them.)

The three challenges among which Rick may choose are also sketched in Fig. 3. Logically, opening the lowest commitment would be a fourth challenge; only it can be executed in parallel with either of the first challenges, thus it has been incorporated into C1.

More precisely, this leads to the following protocol:

First Sybil prepares the commitment structure:

Inner values: Choose R randomly and let $SK' := SK + R$, $PK' = PK{\bullet}g^R$, $sf' = sf + R{*}\underline{m}$ and $s' = sign(SK', m)$. Choose $r \neq 0$ randomly and let $\delta := r(sf' - s')$.

Inner commitments: Compute random commitments b on sf', c on s' (i.e. choose second parameters x, y randomly and apply h), and let $d := (b / c)^r$.

Outer commitments: Finally, compute random commitments A on $\iota(PK')$, B on $\iota(b)$, C on $\iota(c)$, and D on $\iota(d)$.

As sketched in the Fig. 3, the verifier Rick can choose among 3 possibilities:

	Sibyl:		Rick tests if the commitments are opened correctly and if:
C1	Open $A \rightarrow PK'$	Reveal R	$PK' = PK{\bullet}g^R$
	Open $B \rightarrow b$; open $b \rightarrow sf'$		$sf' = sf + R{*}\underline{m}$
	Open $D \rightarrow d$, open $d \rightarrow \delta$		$\delta \neq 0$
C2	Open $A \rightarrow PK'$	Reveal SK'	$PK' = pub(SK')$
	Open $C \rightarrow c$; open $c \rightarrow s'$		$s' = sign(SK', m)$
C3	Open $B \rightarrow b$		
	Open $C \rightarrow c$	Reveal r	$d = (b / c)^r$
	Open $D \rightarrow d$		

Protocol 3 Efficient disavowal

We first show that Protocol 3 is a proof system (in a sense introduced in [BoPe_90], after [ToW1_87]), then that it is perfect zero-knowledge.

Lemma 8: Protocol 3 is an interactive proof system that Sibyl either knows a satisfying assignment for the disavowal formula, or can open a commitment in two ways.

***Proof sketch**:* 1. If Sibyl knows SK and behaves correctly, she can answer all three challenges: The only commitment that she must open and has not chosen explicitly is d. She can open it by showing $\delta = r(sf' - s')$ and $r(x - y)$. The only tests which do not hold by definition are "$\delta \neq 0$?" and "$PK' = pub(SK')$?". The first one follows from $sf' \neq s'$ and $r \neq 0$, the second one from $pub(SK') = pub(SK{+}R) = PK{\bullet}g^R = PK'$.

2. Now we prove that the protocol is convincing. An extractor Eckard lets Sibyl prepare the commitment structure, and then asks her to answer all three challenges. Now Eckard tries to compute SK which satisfies the original formula, or to open a commitment in two ways.

If Sybil herself, in her three answers, opens a commitment in two different ways, the rest is trivial. Otherwise, and if Sibyl can answer correctly in all 3 cases, she outputs R, SK', PK', s', sf', b, c, d, r, δ, x, y, and z such that

$$sf' = sf + R{*}\underline{m} \;\wedge\; PK' = PK{\bullet}g^R \;\wedge\; b = h(sf', x) \tag{17a, b, c}$$

$$d = h(\delta, z) \wedge \delta \neq 0 \tag{17d, e}$$

$$s' = sign(SK', m) \;\wedge\; PK' = pub(SK') \;\wedge\; c = h(s', y) \tag{18a, b, c}$$

$$d = (b / c)^r \tag{19}$$

Let $SK := SK' - R$. Then

$$pub(SK) = pub(SK' - R) = pub(SK') \bullet g^{-R} \underset{18b}{\equiv} PK' \bullet g^{-R} \underset{17b}{\equiv} PK,$$

$$sign(SK, m) = sign(SK', m) - R*\underline{m} \underset{18a}{\equiv} s' - R*\underline{m},$$

and

$$sf \underset{17a}{\equiv} sf' - R*\underline{m}.$$

If $s' \neq sf'$, Eckard has therefore obtained a suitable SK. We therefore show that either $s' \neq sf'$, or Eckard can open the commitment d in two ways. Let $\delta^* := r(sf' - s')$ and $z^* := r(x - y)$. We know

$$h(\delta^*, z^*) = h((sf', x)-(s', y))^r = (h(sf', x)/h(s', y))^r \underset{17c,18c}{\equiv} (b / c)^r \underset{19}{\equiv} d.$$

Thus (δ^*, z^*) is a way of opening the commitment d. If it is different from that in (17d), Eckard succeeds immediately. Otherwise

$$(\delta^*, z^*) = (\delta, z).$$

Together with (17e), this yields

$$0 \neq \delta = \delta^* = r(sf' - s').$$

This implies

$$sf' - s' \neq 0.$$

□

Lemma 9: Protocol 3 is perfect zero-knowledge.

***Proof sketch**:* Like in [Chau1_90], we just show how the three cases can be simulated with the public knowledge only.

(The rest works as in other proofs, e.g. [ToW1_87]: The simulator Simon guesses which case the verifier will choose and simulates it; if he has guessed wrong, he throws the commitment structure away (e.g. erases it from his tape) and chooses a new one. Each time, with probability 1/3, he succeeds. Thus after a period of time about three times as long as it would take in a real conversation, he has a valid looking transcript.)

C1 Choose R randomly and compute $sf' = sf + R*\underline{m}$ and $PK' = PK \bullet g^R$. Also choose $\delta \neq 0$ randomly.
Compute random commitments A on $\iota(PK')$, b on sf', d on δ, B on $\iota(b)$, and D on $\iota(d)$.
Choose a value C randomly (from $\mathcal{G}_{p'}$).

C2 Choose SK' randomly and compute $s' = sign(SK', m)$ and $PK' = pub(SK')$.
Compute random commitments A on $\iota(PK')$, c on s', and C on $\iota(c)$.
Choose values B and D randomly.

C3 Choose b, c, and $r \neq 0$ randomly; let $d = (b / c)^r$; compute random commitments B on $\iota(b)$, C on $\iota(c)$, and D on $\iota(d)$; choose a value A randomly.

One can easily see that in each case, Simon can answer the corresponding challenge. It remains to show that in each case, the joint probability distribution on the values visible to Rick in this case is the same for Simon as for Sibyl:

C1: The values R, PK', sf', A, b, and B (and the second parameters in the commitments A, b, B, and D), are chosen by the same algorithm by Simon and Sibyl. They will be considered as fixed during the choice of the remaining values. D is a random commitment on $\iota(d)$ in both cases. The remaining values in Case C1 are δ, z, d, and C. We will distinguish those chosen by Simon by an asterisk.

Simon chooses $\delta^*\neq 0$ and C^* randomly, and d^* as a random commitment on δ^*, e.g. $d^* := h(\delta^*, z^*)$. Sibyl chooses an additional value $r\neq 0$ randomly, and c as a random commitment on s' (which is fixed by R); thus c is a random number, too. Then she computes $\delta := r(sf' - s')$, $z := r(x - y)$, $d := (b / c)^r$, and C as a random commitment on c, e.g. $C := h(\iota(c), \zeta)$.

We show that for each quadruple $(\delta^*, z^*, d^*, C^*)$ chosen by Simon, there is exactly one triple (r, c, ζ) from which Sibyl obtains the same quadruple, i.e.

$$r(sf' - s') = \delta^*;\ r(x - y) = z^*;\ (b / c)^r = d^*, \text{ and } h(\iota(c), \zeta) = C^*.$$

Uniqueness: r must be $\delta^*(sf' - s')^{-1}$. Now c is uniquely determined as $c = b / d^{*\,r^{-1}}$. Finally, $h(\iota(c), \zeta) = C^*$ has exactly one solution ζ. Existence: For the triple just constructed, it only remains to show $r\neq 0$ and $r(x - y) = z^*$. $r\neq 0$ follows from $\delta^*\neq 0$ and $sf' \neq s'$. Finally, on the one hand, $d^* = h(\delta^*, z^*)$, on the other hand, Sibyl opens d^* as $h(\delta^*, r(x - y))$, and this equation has just one solution.

C2: Simon chooses SK' randomly, and Sibyl as $SK' := SK + R$. However, Sibyl chooses R randomly, and there is exactly one R such that $SK' := SK + R$. Now we fix these choices of SK' and R and show that the remaining values are chosen with equal probabilities. First one easily sees that s' and PK' are the same in both cases. The subsequent choice of A, c, and C (and the choice of the second parameters in them) is identical.

Now Simon chooses B and D randomly, whereas Sibyl chooses them as random commitments on values b and d. According to Lemma 6, this makes no difference.

C3: Simon chooses b, c, and A randomly, and Sibyl as random commitments, which is equal (Lemma 6). The algorithms for the remaining choices are identical. □

It can easily be seen that the proof of Theorem 2 is not much changed by the occurrence of the commitment scheme in Lemma 8.

6.2.4 Notes about non-zero-knowledge disavowal

Although we have a practical zero-knowledge disavowal protocol now, let us note that there would be a way to live without one.

If we accept that knowledge leaks in a disavowal, it might help forging signatures; hence signatures produced after a disavowal should be invalid. This can be done in our scheme, since both the signer Sibyl and the recipient Rick are fixed by the protocol. (Otherwise another honest recipient might not notice that a disavowal has already taken place and accept signatures which are invalid.) Rick is fixed because only he trusts the triple g he has chosen. Of course, we must prevent Sibyl from having a disavowal performed with a wrong signature she has made up herself, and thus making a real signature which she has given to Rick invalid. Thus before a disavowal takes place, we require a normal digital signature from Rick that this is indeed the disavowal he wants.

Protocol: During the key exchange, Rick publishes the triple g, and Sibyl the public key PK_A, like in § 4. Additionally, Rick publishes a public key PK_B of a cryptographically strong conventional cryptosystem, such as GMR [GoMR_88].

Signing and verification are unchanged.

Disavowal: Assume Rick claims that (m, s) is a pair of a message and a signature which he has received from Sibyl, and that Sibyl is forced to verify or disavow it, e.g. in court.

1. Rick must sign this challenge, e.g. as $\mathrm{GMR}_{SK_B}(PK_A, m, s)$.
2. If Sibyl receives this signature, she has two possibilities of disavowal:
   a. She presents a signature by Rick under a different challenge, i.e. $\mathrm{GMR}_{SK_B}(PK_A, m', s')$ with $(m', s') \neq (m, s)$.
   b. She forms $s^* := sign(SK_A, m)$ and, if $s^* \neq s$, she presents s^* . Now she must prove that s^* is indeed a valid signature. This can be done using the verification protocol. (Alternatively, Sibyl could even show SK_A.)

We show that Sibyl is unconditionally, Rick cryptographically secure.

Sibyl's Security:

1. As long as she has no signature from Rick, she only gives one signature and performs verifications. Thus the probability that Rick guesses her real signature is as small as before (Theorem 1). Thus with high probability, in the first disavowal, in the signature $\mathrm{GMR}_{SK_B}(PK_A, m', s')$ that Rick gives, $s' \neq s^* := sign(SK_A, m')$. Hence Sibyl can disavow it using Possibility 2b.
2. From now on, she can disavow every challenge $(m, s) \neq (m', s')$ using Possibility 2a; and if Rick tries the same forgery (m', s') again, she can definitely disavow it again using Possibility 2b.

Rick's Security: Assume he has received a signature (m, s), which Sibyl could verify, and that Sibyl can disavow it later. This means either:

1. Sibyl can show a signature $\mathrm{GMR}_{SK_B}(PK_A, m', s')$ with $(m', s') \neq (m, s)$. Since Rick, if he is honest, would only sign $\mathrm{GMR}_{SK_B}(PK_A, m, s)$, this would mean that Sibyl could break GMR.
2. Sibyl can prove that she knows a SK_A' such that $s^* = sign(SK_A', m) \neq s$. This contradicts Theorem 2.

Invisibility: As long as there is no disavowal, the signatures cannot be shown round, as before. After a disavowal, in many cases, invisibility need not be considered any more anyway:

a) If Rick had the right to force Sibyl to verify or disavow a wrong signature (m', s'), he normally would have had the right, too, to force her to verify or disavow the right one, (m, s). Then Rick could have shown round (m, s) anyway.
b) If Rick tries to attack the invisibility of one signature, (m, s), by having a different one, (m', s'), disavowed, (m, s) is no longer valid.
c) If SK_A were shown during a disavowal, Rick could afterwards produce valid signatures himself. Hence he would have to prove that he had (m, s) before the first disavowal.

However, this scheme does ensure invisibility of (m, s) even after a disavowal, if s^*, the correct signature under the wrong message m', is verified with the verification protocol, without publishing SK_A: s^* is just one signature. Hence, by Sibyl's security, (m', s^*) gives no information about what a correct signature under m is.

Evaluation: One thing is not so ugly as one might fear: No broadcast is needed after the key distribution, i.e. there need not be any global knowledge about if a disavowal has already taken place. Also, disavowals can take place arbitrarily often with different third participants (courts).

The need for a secure conventional cryptosystem is ugly in so far as the only sufficiently efficient provably secure one is GMR based on factoring. Thus we need two different cryptographical assumptions.

The most ugly thing is the key exchange: Rick chooses g, and SK_A depends on it, i.e. the number of keys is quadratic in the number of participants. (This was also the case in the scheme with zero-knowledge disavowal in §4. However, in that case, this problem can be removed if the recipients choose one random g together at the beginning, see § 7. Here, however, the dependence of the key on Rick is an essential requirement.)

7 Efficient extension to many long messages

7.1 Common choice of *g*

Firstly, of course, if there are many participants, they can all use the same group $\mathcal{G}_p$, and each recipient Rick needs to publish just one triple g, which can be used by all signers when signing a message for Rick.

The participants can also choose one triple g whose randomness they all trust (in the cryptographical sense). This is feasible, since g consists of random numbers only, so no general multi-party computation protocol is needed.

Note, however, that in contrast to fail-stop signatures, it is dangerous to use an even more efficient common choice based on a weaker assumption, such as having g chosen by a "trusted" device: With fail-stop signatures, everybody would notice if the device had not been trustworthy after all, and they could choose a new g. Here, however, the recipients could just be cheated.

7.2 Tree authentication

An obvious way to extend the scheme to more than one message is to publish sufficiently many public keys. However, public keys must be reliably distributed at the beginning (which is much more difficult than just sending something), thus they should be short.

Hence the previous public keys PK_i for the i-th signature are used as leaves in tree-authentication, similar to [Merk_80, Merk_82], see Fig. 4.

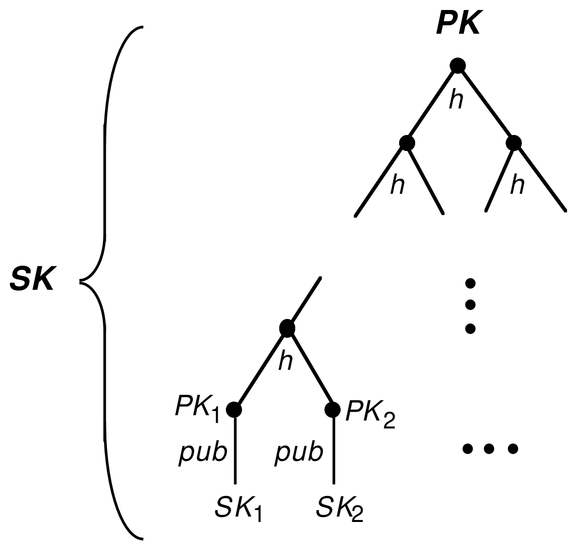

Fig. 4 Authentication tree

The new public key is just the root of the tree. As a secret key, the leaves would suffice, but to save time, one would probably store the whole tree. The values needed to authenticate a specific public key in the old sense (i.e. the values needed before a verification or disavowal protocol can be carried out), are depicted in Fig. 5.

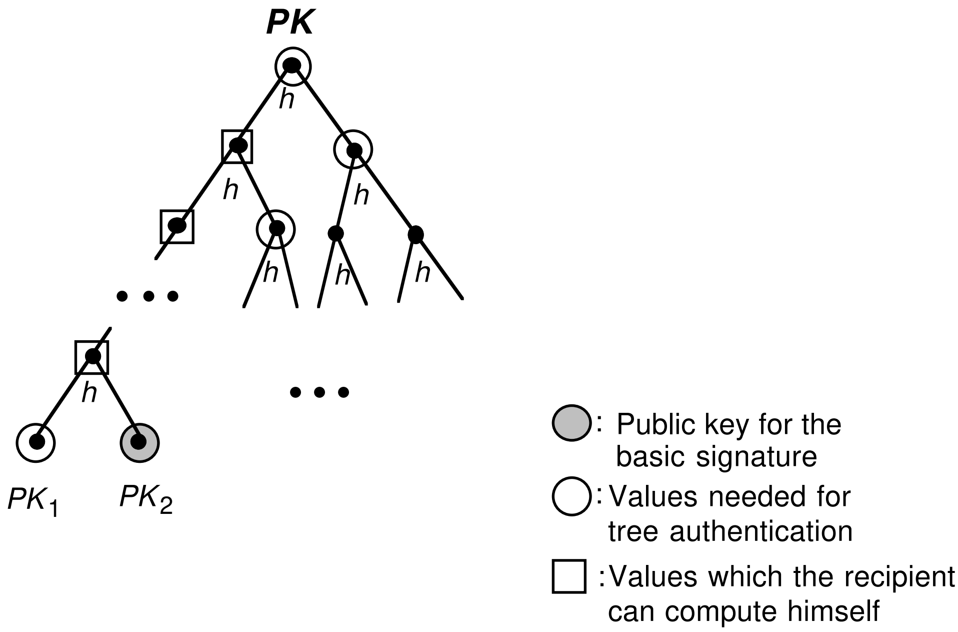

Fig. 5 Authentication of the public key for the second signature

As with fail-stop signatures [Pfit1_89], a collision for the hash function h used for tree-authentication must enable a signer to disavow a signature. (This is done without any additional zero-knowledge protocol: If Sibyl can show such an h-collision, the signature is sufficiently disavowed. Otherwise, the disavowal in the old sense takes place.) Conversely, for the security of the recipient Rick, h must be cryptographically collision-free in the (strong) sense of [Damg_88], and chosen by Rick.

For invisibility, Sibyl must have an opportunity to tell Rick the leaves of the tree. Otherwise, knowing an authenticated leaf of the tree is a strong indication that Rick received it from Alice in a signature. No fault-tolerance is required here. In particular, even if just one g is used (cf. § 7.1), there is no need to broadcast the leaves reliably. It does not even matter if a particular recipient does not receive any leaves, as long as the other participants think he did.

We may want the particular hash functions h used to have three properties:

1. The results should be small, preferably of the size of our public keys.
2. The function should be about as efficient as the rest of the scheme.
3. We may not want to introduce an additional cryptographical assumption for it.

Thus we may not want to use the functions based on factoring used in fail-stop signatures (which were the most efficient ones so far) [Pfit1_89, PfWa_90], or those from [Damg_88]. Those based on the discrete log in [Damg_88] are inefficient, since they need an exponentiation for each bit, whereas our signatures need about a multiplication per bit only.

Thus we should use new ones, based on the fact that n-tuple exponentiation is collision-free (Lemma 3). We do not need hash functions for arbitrarily long messages here (i.e. no hash functions in the sense of [Damg_88] at all), just functions from $D^2 \to D$ for some domain D; called folding functions in [Pfit1_89, PfWa_90].

The obvious idea is to utilize $h^*(x, y) := {g_1}^x \bullet {g_2}^y$, where g_1, g_2 are generators chosen by the recipient. Lemma 3 states that the family of these functions is collision-free. Only their domain and range are not identical, i.e. they are functions $(\mathbb{Z}_p)^2 \to \mathcal{G}_p$. However, Lemma 7 provided efficiently computable bijections ι: $\mathcal{G}_p \to \mathbb{Z}_p$. Thus in these cases, we can use $h := \iota \circ h^*$. (The collision-freeness of h is obvious, since each h-collision is already an h^*-collision because of the injectivity of ι.)

(With elliptic curves, where such bijections are not known, one should use folding functions of more than two parameters. In the general case, we can expect an embedding ι: ${\mathcal{G}_p}^2 \to {\mathbb{Z}_p}^5$. Thus we use a function h^*: ${\mathbb{Z}_p}^5 \to \mathcal{G}_p$ and define $h := h^* \circ \iota$.)

7.3 Message hashing

Long messages can also be hashed before signing. With secure hash functions, this does not make signing much quicker than if one would sign them blockwise, but it saves leaves of the authentication tree. Again, if we have any collision-free hash function (now really in the sense of [Damg_88], i.e. for arbitrarily long messages), we can use it and define that any collision counts as a disavowal.

Again, no previous hash function seems suitable, since they are either based on factoring or inefficient. Thus we should construct one using Lemma 3. We use a tree-like construction again, which can be based on rather general folding functions *fold* (cf. § 7.2). The only problem, compared with the tree-authentication in § 7.2, is that the depth of the tree is not known in advance; thus we must fix it within the construction. (For example, if we have a message at depth 4, the intermediate results at depth 3 are signed, too, thus they must not form a valid message. For a similar reason, a prefix-free encoding had to be used in [GoMR_88, Damg_88].)

Sketch of a simple construction: As a basic folding function, we use $fold(x, y, z) := \iota(g_1{}^x \bullet g_2{}^y \bullet g_3{}^z)$, where g_1, g_2, g_3 are generators chosen by the recipient, and ι is the bijection from Lemma 7. We put the message on one layer (the smallest one where the message fits). The layer with the message is distinguished from all others by starting it with a 1 and all others with a 0 (see Fig. 6).

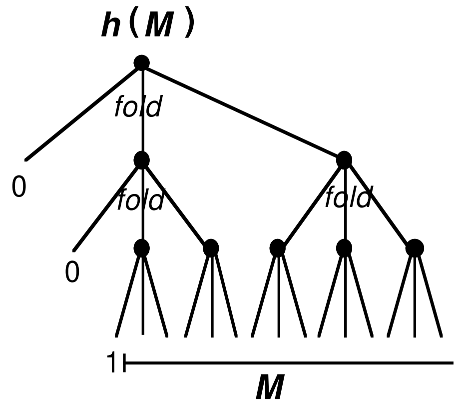

Fig. 6 Simple hash function based on the discrete log

In this construction, on all upper layers some efficiency is lost, since just one bit of the first parameter of the first folding function is used. (This is because the results of the lower layers are exactly of the length of the parameters of this layer.) In the lowest layer, the message can start immediately after the 1-bit. Of course, the value called M in Fig. 6 is not the original message, but a unique encoding, e.g. with a length field and trailing zeroes.

Several parameters can be varied in the construction above. As an example, we present one rather efficient, but specialized, construction in more detail. It assumes a fixed, although very large, upper bound on the message length. (To avoid this, the construction can be iterated or combined with the previous one.)

The idea is that we want a structure like in Fig. 7, where the message is extended over several layers so that there are no unnecessary leaves, and where there is just one bit prefix in each node to indicate whether it is an inner node (0) or a leaf (1). For simplicity, we use binary trees this time, although broader ones may be advantageous, because they have fewer inner nodes.

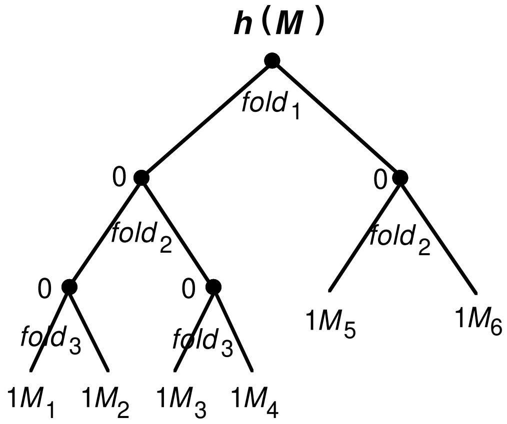

Fig. 7 Example of the second hash function

Construction 2: Assume there are λ layers. A different folding function $fold_i$ must be used in each layer i, so that an element from the range of $fold_i$, prefixed by one bit, is in the domain of $fold_{i-1}$. For example, with the second choice of $\mathcal{G}_p$ from § 3.2, one can choose p_{i-1} two bits longer than p_i. Since the groups $\mathcal{G}_{p_i}$ are the same for all participants and chosen just once, a dozen layers poses no problem here. Of course, one needs a dozen tuples of independently chosen generators, too.

For simplicity, we divide messages into blocks of fixed size $bl = |p_\lambda|-1$. We assume that an encoding has already been applied to the original message (e.g., using trailing zeroes and a length field), so that we are dealing with M of a size $x \bullet bl$. The number of layers needed for this actual M is $d := \lceil ld(x) \rceil$.

We construct a binary tree of depth d with exactly x leaves in the following way: First construct a root, and as its left subtree, a complete tree of depth d–1. Now we have to construct the right subtree with $x' := x - 2^{d-1}$ leaves, and the depth $\lceil ld(x') \rceil < d$. Thus we can apply this procedure recursively, until we have just one leaf left, which is just a root.

Next we distribute M over these x leaves, prefixing each block with a one (cf. Fig. 7, where $x = 6$). Finally, we compute the values of the inner nodes, using the appropriate folding functions, and prefixing each inner node with a 0.

Lemma 10: Construction 2 describes a family h of collision-free hash functions for messages M of length $|M| \le 2^\lambda (|p_\lambda|-1)$.

Proof: It suffices to show that each h-collision (M, M') can be used to compute a collision of one of the folding functions efficiently. Since the tree size is linear in the message length, we have time to search the whole trees T and T' generated during the process of hashing M and M'.

If there is a position where both trees have a node, and the node values differ, consider the branch from the root to this node: Where the node values first differ on this path, there is a collision of a folding function.

Otherwise, since the trees are different, one, e.g. T, must have a node where the other has none. On the path from this node to the root, consider the first position where T' has a node, too. Then that node is an inner node in T, but a leaf in T'. Hence the value in T starts with a 0, that in T' with a 1. This contradicts the assumption of this case. □

7.4 Resulting efficiency

Public keys and signatures are now as short as in conventional signature schemes, such as GMR. In the basic version, keys are specific not only for the signer, but also for the recipient. This, however, can be avoided if a protocol to choose globally trusted random numbers is executed once at the beginning. (And then, there must be some not necessarily reliable broadcast for the leaves of the authentication tree.)

The information exchanged during verification or disavowal is about l times the length of a signature, where l might be between 20 and 100.

Only the secret key SK grows linearly with the number of messages to be signed. This is no great disadvantage, however, since SK is still considerably shorter than the resulting signatures will be, and in most cases, each signature will be stored by at least one participant. Thus if the signatures can be stored, SK can also be stored, particularly since it is only accessed sequentially.

Acknowledgements

We are happy to thank *Manfred Böttger* for initializing this cooperation and for saving us a lot of time by quickly spotting a principle problem in an early version of this scheme, *Matthijs Coster* for his hospitality, *Andreas Pfitzmann* for helpful discussions, and *Michael Waidner* for helpful discussions and tedious proof-reading (but he does not want to be confounded with the readers mentioned in § 3.1).

References

BCDP_90 Joan Boyar, David Chaum, Ivan Damgård, Torben Pedersen: Convertible Undeniable Signatures; Crypto '90, Abstracts, 195-208.

BlPW_90 Gerrit Bleumer, Birgit Pfitzmann, Michael Waidner: A Remark on a Signature Scheme where Forgery can be Proved; Eurocrypt '90, Abstracts, Århus 1990, 403-407.

BoPe_90 Joan Boyar, René Peralta: On the concrete complexity of zero-knowledge proofs; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 507-525.

BrCC_88 Gilles Brassard, David Chaum, Claude Crépeau: Minimum Disclosure Proofs of Knowledge; Journal of Computer and System Sciences 37 (1988) 156-189.

CaWe_79 J. Lawrence Carter, Mark N. Wegman: Universal Classes of Hash Functions; Journal of Computer and System Sciences 18 (1979) 143-154.

ChAn_90 David Chaum, Hans van Antwerpen: Undeniable signatures; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 212-216.

Cha8_85 David Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030-1044.

Chau1_90 David Chaum: Zero-Knowledge Undeniable Signatures; Eurocrypt '90, Abstracts, Århus 1990, 419-426.

ChEG_88 David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf: An improved protocol for demonstrating possession of discrete logarithms and some generalizations; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 127-141.

ChRo_90 David Chaum, Sandra Roijakkers: Unconditionally secure digital signatures; Crypto '90 Abstracts, 209-217.

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.

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.

GoMR_84 Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A "Paradoxical" Solution to the Signature Problem; 25th Symposium on Foundations of Computer Science (FOCS) 1984, 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.

Kobl1_90 Neal Koblitz: Constructing Elliptic Curve Cryptosystems in Characteristic 2; Crypto '90, 11-15 August 1990, Abstracts, 155-168.

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; Westview Press, Boulder 1982, 73-104.

Mil1_86 Victor S. Miller: Use of Elliptic Curves in Cryptography; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 417-426.

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

PfWa_90 Birgit Pfitzmann, Michael Waidner: Formal Aspects of Fail-stop Signatures; Interner Bericht 22/90, Fakultät für Informatik, Universität Karlsruhe, December 1990.

PfWa2_91 Birgit Pfitzmann, Michael Waidner: Fail-stop Signatures and their Applications; accepted for Securicom '91, Paris, March 1991.

RSA_78 Ronald L. Rivest, Adi Shamir, Leonard Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; Communications of the ACM 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96-99.

ToW1_87 Martin Tompa, Heather Woll: Random self-reducibility and zero knowledge proofs of possession of information; 28th FOCS, IEEE Computer Society, 1987, 472-482.

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

WeCa_81 Mark N. Wegman, J. Lawrence Carter: New Hash Functions and Their Use in Authentication and Set Equality; Journal of Computer and System Sciences 22 (1981) 265-279.