Appeared in Proc. 4th International Workshop on Selected Areas in Cryptography (SAC'97), Ottawa, Canada, pp. 231–244, 1997.

Efficient Convertible Undeniable Signature Schemes

(Extended Abstract)

Markus Michels Markus Stadler

Ubilab, UBS
Bahnhofstr. 45
8021 Zurich, Switzerland

Markus.Michels@ubs.com, Markus.Stadler@ubs.com

Abstract

Undeniable signatures are digital signatures which are not universally verifiable but can only be checked with the signer's help. However, the signer cannot deny the validity of a correct signature. An extended concept, convertible undeniable signatures, allows the signer to convert single undeniable signatures or even the whole scheme into universally verifiable signatures or into an ordinary digital signature scheme, respectively.

In this paper we propose a new convertible undeniable signature scheme and provide proofs for all relevant security properties. The scheme is based on Schnorr's signature scheme and it is efficient.

Unlike previous efficient solutions, this new scheme can be used as a basis for an efficient extension to threshold signatures, where the capability of signing (and of verifying signatures) is shared among n parties using a t out of n threshold scheme.

1 Introduction

The two most important properties of ordinary digital signatures are non-repudiation and universal verifiability. Non-repudiation guarantees that a signer cannot deny his or her commitment to a message or a contract at a later time, and the property of universal verifiability allows everybody to check the correctness of a signature. However, for certain applications, universal verifiability is not required or even not desired. Therefore, the concept of undeniable signatures was introduced by Chaum and van Antwerpen [5]. Undeniable signatures are like ordinary digital signatures, with the only difference that they are not universally verifiable. Instead, there exist (often interactive) protocols which allow the signer to convince a verifier about the validity or invalidity of a signature. Non-repudiation is still guaranteed, since the signer cannot convince the verifier that a correct signature is invalid or that an incorrect signature is valid. Various realizations of undeniable signature schemes have been proposed (see [5, 3]). Some concerns about the security of [5] have been discussed in [9, 4, 17]. Moreover, a scheme based on fail stop signatures was suggested [6] and non-interactive undeniable signatures have been introduced [18]. Harn and Yang extended the concept of undeniable signature schemes to a threshold model [14]: the capability of the signer is shared among n parties such that a coalition of at least t parties has to co-operate to sign messages and to verify signatures. They presented schemes for the cases $t = 1$ and $t = n$, however, the latter was successfully attacked by Landau [19]. Recently, Lin, Wang and Chang presented a solution that works with any t, $1 \leq t \leq n$ [20], but it is flawed as well, if signers are not assumed to be honest.

An extended concept of undeniable signatures, called convertible undeniable signatures, was suggested in [2]. With a convertible scheme, the signer can convert undeniable signatures into ordinary, i.e. universally verifiable signatures. This can be done either selectively for single signatures or totally for the whole scheme. Several realizations have been proposed: In [2], a secure but inefficient solution was presented. Practical schemes based on ElGamal signatures [10], which have been proposed in [2] and [23], were shown to be insecure [22], and the solution of [22] lacks detailed security proofs. A scheme proposed by van Heyst and Pedersen [16] can be converted to fail stop signatures, but the key length is linear in the number of signatures that can be signed. Two convertible undeniable signature schemes that are secure w.r.t. forgery have been proposed by Damgård and Pedersen in [8]. These schemes are also based on the ElGamal signature scheme and on techniques for proving that an encrypted signature is valid. One of them uses Rabin-encryption [25] and the interactive verification protocols can be done efficiently. The drawback is that the extension to a threshold scheme is hard to obtain, as a suitable composite modulus must be computed jointly. The second scheme uses ElGamal-encryption, which is somewhat inefficient, as the verification protocol requires several rounds to become secure.

In this paper we present an efficient convertible undeniable signature scheme, which can be proved secure under reasonable cryptographic assumptions. In this scheme, the signer cannot only selectively convert valid signatures into digital signatures, but he or she can also convert any invalid signature into an universally verifiable statement about this fact. The scheme is based on Schnorr's signature scheme [26] and on an efficient zero-knowledge protocol for proving the equality or inequality of discrete logarithms. Furthermore, we show how to extend this scheme to a threshold undeniable signature scheme. Very recently, Gennaro, Krawczyk and Rabin suggested a scheme [13] that is as secure as RSA with respect to forgery. It provides efficient verification protocols but it's less suitable to be a basis of a threshold scheme as a trusted dealer is usually involved to generate the composite RSA-modulus (see [12] for a threshold RSA-scheme) [1].

Our paper is structured as follows: In Section 2 we describe the model of a convertible undeniable signature scheme, then we present a building block which

[1] A way to compute an RSA modulus jointly is suggested in [1], but neither security against an active attacker nor the use of strong primes is guaranteed.

will be used in our protocol. We present our solution in Section 4 and analyze its security. Based on this solution we suggest a threshold scheme in Section 5. Further extensions are discussed in Section 6.

2 Model

A convertible undeniable signature scheme consists of the following procedures.

- A probabilistic set-up algorithm *Setup* which returns the system parameters P.
- A probabilistic key generation algorithm $KeyGen_P$ which, on input the system parameters, returns a key pair (x, y), where x denotes the secret key and y the public key.
- A (possibly probabilistic) signature generation algorithm $SigGen_P(m, x)$ which, on input the secret key x and a message m, returns an undeniable signature s on m.
- A (possibly interactive) verification protocol $Ver_P(m, s, y, x)$ between the signer and the verifier. The signer's input is the secret key x, the message m and the 'alleged' signature s, the verifier's input is m, s and the public key y. The protocol convinces the verifier whether s is a valid signature on m or not.
- A (possibly probabilistic) individual receipt generation algorithm $RecInd_P(m, s, x)$ which, on input a message m, an 'alleged' signature s, and the secret key x, returns an individual receipt r which makes it possible to universally verify whether s is valid or not. A signature can selectively be converted by issuing r.
- An individual verification algorithm $VerInd_P(m, s, y, r)$ which, on input a message m, an 'alleged' signature s, the public key y, and a correct individual receipt r with respect to s, outputs that the receipt r is invalid with respect to s or that r is valid w.r.t. s. If the latter is true, it also outputs whether s is a valid signature on m or not.
- A (possibly probabilistic) universal receipt generation algorithm $RecUni_P(x)$ which, on input the secret key x, returns a universal receipt R which makes it possible to universally verify all signatures. The scheme can be totally converted by releasing R.
- A universal verification algorithm $VerUni_P(m, s, y, R)$ which, on input of a message m, an 'alleged' signature s, the public key y, and a universal receipt R, outputs the the receipt R is invalid or not. If the latter is true is also outputs, whether s is either a valid or a invalid signature on s.

The following statements must hold for a secure undeniable signature scheme:

- *Unforgeability:* The signature scheme is existentially unforgeable under an adaptive attack, i.e., there is no efficient algorithm that returns a valid signature s on an arbitrary message m with non-negligible probability, even if a polynomial number of valid signatures on chosen messages are given.

- *Invisibility:* There exists no efficient algorithm which, on input the public key y, a message m, and an 'alleged' signature s, can decide with non-negligible probability better than guessing whether s is either valid or not.

- *Completeness and soundness of verification:* The verification algorithms *Ver*, *VerInd* and *VerUni* are complete and sound, where completeness means that valid (invalid) signatures can always be proved valid (invalid), and soundness means that no valid (invalid) signature can be proved invalid (valid). Indirectly, this must also hold for the algorithms *RecInd*, *RecUni*.

- *Non-transferability:* A verifier participating in an execution of the interactive verification *Ver* of a signature does not obtain information that could be used to convince a third party about the correctness of a signature (although this verifier knows whether the given signature is valid or not).

3 Preliminaries and Building Blocks

In this section we first describe briefly the notation we use. Then we present an efficient interactive zero-knowledge proof for showing that two discrete logarithms are either equal or not. This protocol will be used later in our scheme, but is of independent interest. Such a proof is also called a *biproof* [11], because it proves that the input word belongs to one of two languages. A less efficient bit-wise proof for this problem has been suggested in [11].

3.1 Notations

$\mathbf{Z}_q$ denotes the ring of integers modulo q and $\mathbf{Z}_p^*$ denotes the multiplicative group modulo p. We write $a \in_R \mathcal{A}$ to indicate that the value a is chosen randomly from the set $\mathcal{A}$ according to the uniform distribution.

In the sequel, we will make use of a cyclic group $G = \langle \alpha \rangle$ of prime order q, in which computing discrete logarithms is infeasible. For instance, G could be constructed as a subgroup of the group $\mathbf{Z}_p^*$ for a suitable prime with $q|(p-1)$, or G could be an elliptic curve.

We also assume collision resistant hash functions $\mathcal{H}_\ell : \{0,1\}^* \times G \rightarrow \{0,1\}^\ell$ (with $\ell = O(\log_2 q)$, in a practical realization e.g. $\ell \approx 160$), the hash function family $\mathcal{H}_t : G^t \rightarrow \{0,1\}^\ell$ and $\mathcal{H}_G : \{0,1\}^* \rightarrow G$. If $G \subset \mathbf{Z}_p^*$, the latter could for instance be constructed by first hashing to a string of length $\log_2 p$ and then computing the $((p-1)/q)$-th power of this value.

3.2 Proving the equality or inequality of two discrete logarithms

An important component of our realization of a convertible undeniable signature scheme is an efficient protocol that allows a prover to convince a verifier about the equality or inequality of two discrete logarithms, such that no additional information about these logarithms is leaked. More precisely, assume the prover knows the discrete logarithm x of $y = \alpha^x$ and wants to allow the verifier to decide whether $\log_\beta z = \log_\alpha y$ for given group elements β and z. Therefore, the prover and the verifier execute the following protocol.

1. The verifier chooses random values $u, v \in \mathbf{Z}_q$, computes $a := \alpha^u y^v$, and sends a to the prover.

2. The prover chooses random values $k, \tilde{k}, w \in \mathbf{Z}_q$, computes $r_\alpha := \alpha^k$, $r_\beta := \beta^k$, $\tilde{r}_\alpha := \alpha^{\tilde{k}}$, and $\tilde{r}_\beta := \beta^{\tilde{k}}$, and sends r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, and w to the verifier.

3. The verifier opens his commitment a by sending u and v to the prover.

4. If $a \neq \alpha^u y^v$ the prover halts, otherwise he computes $s := k - (v + w)x \pmod{q}$, $\tilde{s} := \tilde{k} - (v + w)k \pmod{q}$ and sends s and $\tilde{s}$ to the verifier.

5. The verifier first checks whether $\alpha^s y^{v+w} = r_\alpha$, $\alpha^{\tilde{s}} r_\alpha^{v+w} = \tilde{r}_\alpha$, and $\beta^{\tilde{s}} r_\beta^{v+w} = \tilde{r}_\beta$ and then concludes:

$$\begin{cases} \text{if } \beta^s z^{v+w} = r_\beta & \text{then } \log_\beta z = \log_\alpha y \\ \text{if } \beta^s z^{v+w} \neq r_\beta & \text{then } \log_\beta z \neq \log_\alpha y \end{cases}$$

The following theorem can be proved.

Theorem 1 *The above protocol is complete and sound. It is zero-knowledge under the assumption that there exists no algorithm running in expected polynomial time which decides with non-negligible probability better than guessing whether two discrete logarithms are equal.*

Proof (Sketch): The completeness of the protocol is obvious because an honest prover can always convince an honest verifier of the equality or inequality of the two discrete logarithms. To prove the soundness property, it is important to note that the commitment a sent by the verifier in the first message does not reveal any information (in an information-theoretic sense) about the value v and that therefore the "challenge" $v + w$ is truly random for the prover. Based on this observation it can easily be shown that successful cheating is only possible with negligible probability.
To prove the zero-knowledge property of the protocol, we show how to construct a simulator that returns a protocol transcript with a probability distribution indistinguishable from the distribution of a verifier's protocol view. The simulator uses the verifier as a black-box, i.e., it works independently from the verifier's strategy.

1. The verifier's algorithm is used to compute the commitment a.

2. The simulator randomly chooses s, $\tilde{s}$, w, and $c \in \mathbf{Z}_q$ and computes $r_\alpha := \alpha^s y^c$, $\tilde{r}_\alpha := \alpha^{\tilde{s}} r_\alpha^c$, and $\tilde{r}_\beta := \beta^{\tilde{s}} r_\beta^c$. The value r_β is computed as $\beta^s z^c$ or chosen randomly from $G \backslash \{\beta^s z^c\}$, depending on which protocol outcome should be simulated.

3. The verifier's algorithm is used to compute u and v on input the values r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, and w.

4. If $a \neq \alpha^u y^v$ the simulator returns as protocol transcript the values r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, w, and `halt`. Otherwise, the simulator repeats steps 2 and 3 until the commitment a is correctly opened with values u' and v'; step 2 is modified such that c is not chosen randomly but set to $c = v + w$.

5. If the simulator finally stops and if $u = u'$, it returns as transcript the values r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, w, s and $\tilde{s}$. If $u \neq u'$ then the discrete logarithm x of y to base α can be extracted and the simulator returns as transcript the value of x.

It can easily be seen that the expected number of repetitions of steps 2 and 3 is constant and that therefore the simulator runs in expected polynomial time. Let us now explain briefly why the output of the simulator is computationally indistinguishable from a protocol view:

- the probability that a `halt` occurs is the same as in a protocol execution.
- the probability that the simulator returns x, the discrete logarithm of y to the base α, is negligible because of the assumption (otherwise, the simulator could be used to test the equality of discrete logarithms with non-negligible probability).
- all values, except $\tilde{r}_\beta$, are distributed according to the same distribution in the simulator's output and the protocol view, and the two distributions of $\tilde{r}_\beta$ can distinguished only by deciding whether $\log_\alpha \tilde{r}_\alpha$ equals $\log_\beta \tilde{r}_\beta$, which is not possible according to the assumption.

□

To obtain a designated verifier proof [18], the commitment a must be computed as $a := \alpha^u y_V^v$, where y_V is the verifier's public key and $w = \mathcal{H}_{10}(\alpha, y, \beta, z, r_\alpha, r_\beta, \tilde{r}_\alpha, \tilde{r}_\beta, a, y_V)$. This proof is non-interactive, but as the verifier can generate this proof as well, it's not a receipt.

The protocol can also easily be turned into a non-interactive argument by omitting the commitment a, setting w to 0 and computing v as $v = \mathcal{H}_8(\alpha, y, \beta, z, r_\alpha, r_\beta, \tilde{r}_\alpha, \tilde{r}_\beta)$.

4 New convertible signature scheme

We describe our scheme and analyze its security.

4.1 Basic scheme

The protocol can be described as follows:

1. *Set up:* The system parameters are G, α, q, $\mathcal{H}_\ell$, and $\mathcal{H}_G$.

2. *Key generation:* Each user picks at random two numbers x_1 and x_2 from $\mathbf{Z}_q$ as secret keys and computes the public keys $y_1 := \alpha^{x_1}$ and $y_2 := \alpha^{x_2}$.

3. *Signature generation:* A message m is signed in the following way:

   (a) $k \in_R \mathbf{Z}_q$, $r := \alpha^k$, $\tilde{r} := \mathcal{H}_G(r)^{x_2}$

   (b) $c := \mathcal{H}_\ell(m, \tilde{r})$

   (c) $s := k - cx_1 \pmod q$

   The resulting signature on m is the pair $(\tilde{r}, s)$.

4. *Interactive verification:* The signature can be verified or denied by interactively proving the equality or inequality of the discrete logarithms of $\tilde{r}$ and y_2 to the bases $\mathcal{H}_G(\alpha^s y_1^{\mathcal{H}_\ell(m,\tilde{r})})$ and α, respectively, using the interactive protocol described in section 3.2. Alternatively, the non-interactive designated verifier proof outlined in section 3.2 can be used.

5. *Individual receipt generation:* To selectively convert a signature, this proof is turned into an non-interactive argument using the non-interactive argument protocol described in section 3.2. This argument is the individual receipt. Note that this also allows to make it publicly verifiable that a given signature is *invalid.*

6. *Individual verification:* By checking the validity of the individual receipt, a verifier can see whether the related signature is either valid or invalid.

7. *Universal receipt generation:* In order to totally convert all undeniable signatures into digital signatures, the secret key x_2 is published as universal receipt.

8. *Universal verification:* The verifier checks whether

$$\mathcal{H}_G(\alpha^s y_1^{\mathcal{H}_\ell(m,\tilde{r})})^{x_2} \equiv \tilde{r}$$

   holds.

4.2 Efficiency

We analyse the efficiency of our scheme, where G is chosen as the multiplicative subgroup of order q of Z_p^* and q is small. Thus in general we have short exponents. Only for the computation of $\mathcal{H}_G$ we need a full exponentiation, as we exponentiate an output of a hash function $h : \{0,1\}^* \rightarrow Z_p$ with $(p-1)/q$ to get an element in G. Let $M_l(i)$ denote the number of $|p|$-bit multiplications that are required to compute i cascaded exponentiations of the form $\prod_{j=1}^{i} r_j^{d_j}$

Operations	Signer	Verifier
Signature generation	$2 \cdot M_Q(1) + M_{P-Q}(1)$	–
Interactive verification	$5 \cdot M_Q(1) + M_{P-Q}(1)$	$4 \cdot M_Q(2) + M_{P-Q}(1)$
Selective conversion	$5 \cdot M_Q(1) + M_{P-Q}(1)$	–
Individual verification	–	$4 \cdot M_Q(2) + M_{P-Q}(1)$
Total conversion	–	–
Universal verification	–	$M_Q(1) + M_Q(2) + M_{P-Q}(1)$

Figure 1: Costs for operations

where l is the length of the exponents. Let $P = |p|$ and $Q = |q|$. Figure 1 shows the costs for the different operations.

Let us illustrate the costs in an example with $P = |p| = 1024$ and $Q = |q| = 256$. Using methods of [29], we have $M_P(1) = 308, M_Q(2) = 373$ multiplications for an exponentiation with one and two 256-bit exponents, respectively and $M_{P-Q}(1) = 902$ multiplications for an exponentiation with one 768-bit exponent. The costs for the different operations in this example are described in Figure 2.

Operations	Signer	Verifier
Signature generation	1518	–
Interactive verification	2442	2394
Selective conversion	2442	–
Individual verification	–	2394
Total conversion	0	–
Universal verification	–	1583

Figure 2: Costs in number of 1024-bit multiplications

Even better results can be achieved if G is chosen to be an elliptic curve over a finite field.

4.3 Security analysis

We distinguish the analysis in the parts unforgeability, invisibility, untransferability and completeness & soundness of verification.

Unforgeability

Theorem 2 *Under the assumption that the hash functions $\mathcal{H}_\ell$ and $\mathcal{H}_G$ are truly random functions, forging valid signatures is equivalent to forging Schnorr signatures.*

Proof (Sketch): In the converted scheme, with known x_2, let the function $\mathcal{H}_N : \{0,1\}^* \times G \to \{0,1\}^\ell$ be defined as

$$\mathcal{H}_N(m, r) := \mathcal{H}_\ell(m, \mathcal{H}_G(r)^{x_2}).$$

The converted undeniable signature scheme is equivalent to a Schnorr signature scheme using the hash function $\mathcal{H}_N$. But because $\mathcal{H}_N$ is indistinguishable from a truly random function (the only differences are possible collisions of $\mathcal{H}_G$), the converted signature scheme is secure. As a consequence, also the non-converted scheme has to be secure. □

Invisibility

To prove the invisibility of the scheme prior to conversion, we need an additional assumption, the Decision Diffie-Hellman assumption. We first define the two sets

$$\mathcal{DH} := \{(\alpha, y, \beta, z) \in G^4 \mid \log_\alpha y = \log_\beta z\}$$

$$\mathcal{NDH} := \{(\alpha, y, \beta, z) \in G^4 \mid \log_\alpha y \neq \log_\beta z\}$$

of Diffie-Hellman and non-Diffie-Hellman 4-tuples.

Assumption For all probabilistic polynomial time algorithms $\mathcal{A} : G^4 \to \{0, 1\}$, the two probability distributions

$$\mathrm{Prob}_{T \in_R \mathcal{DH}}\,[\mathcal{A}(T) = 1] \quad \text{and} \quad \mathrm{Prob}_{T \in_R \mathcal{NDH}}\,[\mathcal{A}(T) = 1]$$

are computationally indistinguishable (the probabilities are taken over the random coin tosses of $\mathcal{A}$ and over the random choices from $\mathcal{DH}$ and $\mathcal{NDH}$, respectively).

Theorem 3 Provided that the assumption holds and that the the hash function $\mathcal{H}_G$ can't be distinguished from a random function, verifying a given signature without the assistance of the signer can be done only with negligible probability, even if a polynomial number of valid signatures is known.

Proof (Sketch): The basic idea is to transform an instance (α, y, β, z) of the above problem into an instance of the signature scheme (we assume that the instance (α, y, β, z) is already randomized, e.g. see [28]). Concretely, let α be the generator, let y be y_2, and let $y_1 = \alpha^{x_1}$ for a randomly chosen x_1. Furthermore, we simulate the hash function $\mathcal{H}_G$ to be able to generate signatures. To generate a correct signature, we proceed as described, except that we set $\mathcal{H}_G(r) = \alpha^t$ and $\tilde{r} = y_2^t$ for a randomly chosen t (this guarantees that the signature is correct, even if we don't know x_2). For the signature whose verification is to be equivalent to solving the above problem, let $\mathcal{H}_G(r) = \beta$ and $\tilde{r} = z$: this signature is only valid if the above 4-tuple is a Diffie-Hellman tuple. Therefore, if there was any efficient algorithm which can decide (better than guessing) whether this signature is valid, such an algorithm could also be used to solve the Decision-Diffie-Hellman problem, but this would lead to a contradiction.

Non-transferability

Non-transferability follows directly from the zero-knowledge property of the interactive protocol for proving the equality/inequality of discrete logarithm.

Soundness & completeness of verification

Before conversion, these properties follow from the soundness and the completeness property of the used zero-knowledge protocols. In the selectively converted version, these properties are inherited because of the impossibility to issue wrong receipts provided the used hash function is collision resistant.

5 Robust convertible undeniable threshold signature scheme

Using standard techniques for verifiable sharing of discrete logarithms [23] and methods from secure multi-party computations, our basic convertible undeniable signature scheme presented in Section 4 can be adapted for the threshold scenario.

5.1 Model

In a convertible undeniable threshold signature scheme there is a group of n signers such that any coalition of at least t signers can jointly sign a message.

The communication model is as follows: During the signature generation, it is assumed that the signers can broadcast messages to each other and the signers check proofs of other signers. During the interactive verification each signer has a channel to communicate to the verifier.

The threshold scheme consists of the same procedures as listed in section 2, however, the key generation algorithm must output shares of the secret key for each signer such that only at least t signer are able to sign a document on behalf of the group, and the other algorithms should be adopted accordingly.

With regard to security against dishonest signers, we have to distinguish between passive and active cheaters. Passive cheaters follow the protocols honestly but try to gain additional knowledge by pooling their information, while active cheaters can even deny service or send wrong values.

5.2 New scheme

We outline the robust convertible undeniable threshold signature scheme, based on the scheme given in the previous section.

It is assumed that x_1 and x_2 are shared among n provers using a verifiable t out of n threshold secret sharing scheme (for details see [27, 23]). A share of signer i of a variable or value a is denoted $Share_i(a)$. Given at least t distinct (correct) shares, the value of a can be reconstructed using Lagrange interpolation (see [23]). We also assume that shadows of the form $\alpha^{Share_i(x_1)}$ are publicly known.

A message m is signed in the following way by d signers ($t \leq d \leq n$):

1. The signers jointly compute $r := \alpha^k$ in a distributed manner. Each signer i gets a verifiable share $Share_i(k)$. A shadow $\alpha^{Share_i(k)}$ is revealed and publicly known.

2. Each signer computes $\tilde{r_i} := \mathcal{H}_G(r)^{Share_i(x_2)}$ and proves interactively and in zero-knowledge to all other signers that this is correctly done. This requires only a simple zero knowledge proofs of equality of discrete logarithms. If at least t signers are honest, each of them can compute $\tilde{r} = \mathcal{H}_G(r)^{x_2}$ by combining the values $\tilde{r_i}$ of the honest signers.

3. Each signer computes $c := \mathcal{H}_\ell(m, \tilde{r})$.

4. Signer i computes $Share_i(s) := Share_i(k) - c \cdot Share_i(x_1) \pmod q$ and broadcasts this value to all other signers. These shares are checked using the revealed shadows. If at least t signers are honest, the value s can be reconstructed, which is then sent to the verifier.

The verification protocols, as well as the procedures for generating receipts, can be adapted from the basic protocols described in Section 4 in an analogous way.

5.3 Security analysis

For an outsider attacker the security analysis does not differ from the analysis given in the previous section.

As d signers ($t \le d \le n$) sign a message, we have to assume that there are at most $t-1$ dishonest signers and among those, there must not be more than $min(d-t, t-1)$ actively cheating signers. We further have to assume that the verifier has no unconditional trust to any signer.

- *Key Generation:* It was already shown that any group of $t-1$ members does not obtain any knowledge concerning the secret keys and it's impossible to cheat for signers during the key generation protocol [23].

- *Unforgeability:* The dishonest signer could pick a document m and try to get a threshold signature on it, although the honest signers are not aware of this document (e.g. they might think to sign another document m'). Such an attack was successful in some multi party signature schemes as pointed out in [21]. Here, however, such an attack is not successful. It is impossible to transform a partial signature of a honest signer on m' to a partial signature on m, as r can't be fully determined by the dishonest signer and r and m' are both input of the hash function.

- *Invisibility:* Invisibility still holds even if $t-1$ signers send the knowledge they gained during signature generation to the verifier provided the verifier does not trust any signer unconditionally. In fact, r is known by the verifier anyway, but some $\mathcal{H}_G(r)^{Share_i(x_2)}$ and the partial signatures can be send him as extra knowledge. However, the relation $\tilde{r} = \mathcal{H}_G(r)^{x_2}$ can't be proved by $t-1$ signers. The partial signatures are useless as well, as they can be simulated by one signer.

- *Non-transferability:* As the interactive verification is zero-knowledge and the information from selective conversion for given signatures does not help to verify another alleged signature, non-transferability holds.
- *Robustness:* As only up to $min(d-t, t-1)$ signers are totally controlled by the attacker, the signature can always be generated by the t remaining signers, that are either honest or only passively cheating.

6 Further extensions

Clearly, the security model can be somewhat strengthend by updating the individual shared parameters from time to time without changing the public key [15]. Furthermore, the threshold schemes can be transformed into shared signature schemes with arbitrary access structure just by substituting the used secret sharing scheme. It's also possible to share the verifier by using ideas proposed in [23, 18] or to use publicly verifiable secret sharing [28] instead of the non-publicly verifiable sharing scheme.

References

[1] D.Boneh, M.Franklin, "Efficient generation of shared RSA keys", LNCS, Proc. Crypto'97, Springer Verlag, (1997).

[2] J.Boyar, D.Chaum, I.Damgård, T.Pedersen, "Convertible undeniable signatures", LNCS 537, Proc. Crypto '90, Springer Verlag, (1991), pp. 189–205.

[3] D.Chaum, "Zero-knowledge undeniable signatures", LNCS 473, Proc. Eurocrypt '90, Springer Verlag, (1991), pp. 458–464.

[4] D.Chaum, "Some weakness of "Weaknesses of Undeniable Signatures"", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 554–556.

[5] D.Chaum, H. van Antwerpen, "Undeniable Signatures", LNCS 435, Proc. Crypto '89, Springer Verlag, (1990), pp. 212–216.

[6] D.Chaum, E. van Heyst, B.Pfitzmann, "Cryptographically strong undeniable signatures, unconditionally secure for the signer", LNCS 576, Proc. Crypto'91, (1992), pp. 470–484.

[7] D.Chaum, T.Pedersen, "Wallet databases with observers", LNCS 740, Proc. Crypto'92, (1993), pp. 89 – 105.

[8] I.Damgård, T.Pedersen, "New convertible undeniable signature schemes", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 372–386.

[9] Y.Desmedt, M.Yung, "Weaknesses of undeniable signature schemes", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 205–220.

[10] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", LNCS 196, Proc. Crypto'84, (1985), pp. 10–18.

[11] A.Fujioka, T.Okamoto, K.Ohta, "Interactive Bi-Proof Systems and undeniable signature schemes", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 243–256.

[12] R.Gennaro, S.Jarecki, H.Krawczyk, T.Rabin, "Robust and efficient sharing of RSA functions", LNCS 1109, Proc. Crypto'96, Springer Verlag, (1996), pp. 157–172.

[13] R.Gennaro, H.Krawczyk, T.Rabin, "RSA-based undeniable signatures", LNCS, Proc. Crypto'97, Springer Verlag, (1997).

[14] L.Harn, S.Yang "Group-oriented undeniable signature schemes without the assistance of a mutually trusted party", LNCS 718, Proc. Asiacrypt'92, Springer Verlag, (1993), pp. 133–142.

[15] A.Herzberg, S.Jarecki, H.Krawczyk, M.Yung, "Proactive secret sharing or: How to cope with perpetual leakage", LNCS 963, Proc. Crypto'95, Springer Verlag, (1995), pp. 339–352.

[16] E.van Heyst, T.Pedersen, "How to make efficient fail stop signatures", LNCS 658, Proc. Eurocrypt'92, Springer Verlag, (1993), pp. 366–377.

[17] M.Jakobsson, "Blackmailing using undeniable signatures", LNCS 950, Proc. Eurocrypt'94, Springer Verlag, (1995), pp. 425–427.

[18] M.Jakobsson, K.Sako, R.Impagliazzo, "Designated verifier proofs and their applications", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 143–154.

[19] S.Landau, "Weaknesses in some threshold cryptosystems", LNCS 1109, Proc. Crypto'96, Springer Verlag, (1996), pp. 74–83.

[20] C.-H.Lin, C.-T.Wang, C-C.Chang, "A group oriented (t,n) undeniable signature scheme without trusted center", LNCS 1172, Information Security and Privacy, ACISP'96, Springer Verlag, (1996), pp. 266–274.

[21] M.Michels, P.Horster, "On the risk of disruption in several multiparty signature schemes", LNCS 1163, Proc. Asiacrypt '96, Springer Verlag, (1996), pp. 334–345.

[22] M.Michels, H.Petersen, P.Horster, "Breaking and repairing a convertible undeniable signature scheme", Proc. 3rd ACM Conference on Computer and Communications Security, ACM Press, (1996), pp. 148–152.

[23] T.P.Pedersen, "Distributed provers with applications to undeniable signatures", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 221–242.

[24] D.Pointcheval, J.Stern, "Security proofs for signature schemes", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 387–398.

[25] M.O.Rabin, "Digitalized signatures and public-key functions as intractable as factorization", MIT/LCS/TR-212, MIT Lab. for Computer Science, Cambridge, Mass., 1979.

[26] C.P.Schnorr, "Efficient signature generation for smart cards", Journal of Cryptology, Vol. 4, (1991), pp. 161–174.

[27] A.Shamir, "How to share a secret", Communications of the ACM, Vol.22, (1979), pp. 612–613.

[28] M.Stadler, "Publicly verifiable secret sharing", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 190–199.

[29] S.-M.Yen, C.-S.Laih, "The fast cascade exponentiation algorithm and its applications on cryptography", LNCS 718, Proc. Asiacrypt'92, Springer Verlag, (1993), pp. 447–456.