# Generic Constructions for Secure and Efficient Confirmer Signature Schemes

**(Extended Abstract)**

Markus Michels and Markus Stadler

Ubilab, UBS
Bahnhofstr.45
8021 Zurich, Switzerland
`Markus.Michels@ubs.com` `Markus.Stadler@ubs.com`

**Abstract.** In contrast to ordinary digital signatures, the verification of undeniable signatures and of confirmer signatures requires the cooperation of the signer or of a designated confirmer, respectively. Various schemes have been proposed so far, from practical solutions based on specific number-theoretic assumptions to theoretical constructions using basic cryptographic primitives. To motivate the necessity of new and provably secure constructions for confirmer signatures, we first describe a flaw in a previous realization by Okamoto. We then present two generic constructions for designing provably secure and efficient confirmer variants of many well-known signature schemes, including the schemes by Schnorr, Fiat and Shamir, ElGamal, and the RSA scheme. The constructions employ a new tool called confirmer commitment schemes. In this concept the ability to open the committed value is delegated to a designated confirmer. We present an efficient realization based on the Decision-Diffie-Hellman assumption.

**Keywords:** designated confirmer signature schemes, undeniable signature schemes, commitment schemes, provable security.

## 1 Introduction

Digital signatures are an important tool for realizing security in open distributed systems and in electronic commerce as they guarantee the authenticity of data. In the common model, a digital signature can be verified by everyone (universal verifiability) and therefore its validity cannot be denied by the signer (non-repudiation). However, in certain applications universal verifiability is not desirable, for instance if a provable commitment presents valuable information, like in an auction with signed bids. *Undeniable signatures*, introduced by Chaum and van Antwerpen in [7], are a solution to this problem. Such signatures can be verified only with the signer's cooperation in a way that preserves the non-repudiation property. This implies that a verifier cannot check the validity of a signature on his own. This property is called *invisibility* or simulatability. Practical realizations of undeniable signature schemes have been proposed in

[7] and [4]. Other solutions have the additional property that the scheme can later be converted into an ordinary signature scheme with universal verifiability (e.g. [3, 27, 21, 11, 22, 16]). While all these solutions are based on particular number theoretic problems, Boyar et al. [3] show how to construct a convertible undeniable signature scheme based on any digital signature scheme.

A drawback of (convertible) undeniable signatures is that the signer could be forced to cooperate in the verification of a signature. *Entrusted undeniable signatures* [28] overcome this problem. However, a more severe problem is that the verification of signatures is not possible if the signer is absent or denies collaboration. This problem can be overcome with *designated confirmer signatures* [6]. In this scenario, the ability to verify signatures is delegated to an additional entity, called the confirmer. Concrete realizations were presented in [6, 26]. Furthermore, Okamoto showed in [26] that confirmer signature schemes exist if and only if public key encryption schemes exist.

**Our Contribution:** Several of the previously proposed undeniable signature schemes have shown to be flawed, as was described in [12, 20, 21], and it is possible to show that the entrusted undeniable signature scheme described in [28] is flawed as well. Here we point out in detail that the confirmer signature schemes by Okamoto [26] have a certain weakness as well.

The main result of this paper consists of two generic constructions for confirmer signatures. Unlike the constructions for undeniable and confirmer signatures presented in [3] and [26], respectively, which are merely proofs of existence and result in inefficient solutions, our constructions yield efficient schemes (the practical construction proposed in [26] is insecure, see Section 3). We introduce a new tool named *confirmer commitments*. Compared to ordinary commitment schemes, confirmer commitments are not opened by the commiter but by a designated confirmer. Moreover, the confirmer is able to prove efficiently whether or not a given commitment "contains" a certain message without revealing any useful knowledge. We describe a concrete realization whose security is equivalent to the Decision-Diffie-Hellman assumption and discuss an alternative construction based on the higher residue assumption. Using such a confirmer commitment scheme and a secure signature scheme satisfying certain conditions, our constructions yield secure confirmer signature schemes. Possible candidates for the underlying signature schemes are, among others, the schemes by Schnorr [31], Fiat-Shamir [14], ElGamal [13], and the RSA scheme [30]. Our constructions have the interesting feature that it is possible to choose the commitment scheme and the signature scheme, as well as the security parameters, independently. With respect to unforgeability, the security of the resulting confirmer scheme is equivalent to the security of the underlying signature scheme, given that the confirmer commitment is collision resistant. Whereas with respect to invisibility, its security is equivalent to the security of the confirmer commitment scheme. Finally, we outline how to deal with multiple confirmers.

**Organization of the Paper:** Section 2 contains our model of confirmer signature schemes. In Section 3 a previously proposed scheme is analyzed and a weakness is pointed out. Then we introduce the concept of confirmer commitments in Section 4. Section 5 describes how to design secure confirmer signature schemes based on confirmer commitments and proof systems or existentially forgeable signature schemes. Some extensions are discussed in Section 6.

## 2 Model

The players in a designated confirmer signature scheme are a signer $S$, a confirmer $C$, and a verifier $V$. Note that it is possible that the signer also plays the role of the confirmer, in which case the scheme becomes an "ordinary" undeniable signature scheme. A confirmer signature scheme consists of the following efficient algorithms and protocols:

- Two key generators $KG_S(1^\ell) \rightarrow (x_S, y_S)$ and $KG_C(1^\ell) \rightarrow (x_C, y_C)$. The parameter $\ell$ is a security parameter, $(x_S, y_S)$ is a secret/public key pair for the signer and $(x_C, y_C)$ is a secret/public key pair for the confirmer.
- A probabilistic signature generation algorithm $Sig(m, x_S, y_C) \rightarrow \sigma$ for signing a message $m \in \{0,1\}^*$.
- An interactive signature verification protocol $\langle C_{Ver}, V_{Ver} \rangle$ between the confirmer and the verifier:

  $$\langle C_{Ver}(x_C), V_{Ver}() \rangle (m, \sigma, y_S, y_C) \rightarrow_V \begin{cases} 1 \\ 0 \end{cases} .$$

  The private input of $C_{Ver}$ is $x_C$ and the common input consists of $m$, $\sigma$, $y_S$, and $y_C$. After a successful protocol execution the verifier's output (indicated by $\rightarrow_V$) is either 1 or 0. Note that in some realizations there exist separate protocols for confirming and denying signatures. However, these two protocols can easily be merged into a single protocol.

*Remarks:* In this model, both confirmation and disavowal are delegated to the confirmer. However, the signer is always able to confirm (but not necessarily to deny) a signature generated by himself by proving his knowledge of the secret key and of the random values used for generating this signature. In certain applications it could be useful that the signer also knows the key $x_C$ or if a scheme with multiple confirmers is used (see Section 6).

Moreover, the confirmer is able to (totally) convert the scheme into an ordinary signature scheme by releasing his secret key $x_C$. If there is also an efficient non-interactive variant of the proof $\langle C_{Ver}, V_{Ver} \rangle$ (called *receipt*), he is able to convert single signatures as well.

The following definitions are required for the definitions of the security properties of confirmer signature schemes.

- *Correctness:* A message-signature pair $(m, \sigma)$ is defined to be *correct* (with respect to $y_S$ and $y_C$) if and only if $\Pr[Sig(m, x_S, y_C) = \sigma] > 0$. In other words, $(m, \sigma)$ is correct if $\sigma$ could have been generated by $Sig(m, x_S, y_C)$.

- *Oracles:* For fixed keys $x_S$, $y_S$ and $y_C$ let $O_S$ be an oracle which on input $m$ returns a signature $\sigma$ generated according to the probability distribution of $Sig(m, x_S, y_C)$ and let $O_V$ be an oracle which on input a message-signature pair $(m, \sigma)$ outputs whether or not $(m, \sigma)$ is correct with respect to $y_S$ and $y_C$. Let further $M_S$ and $M_V$ denote the sets of messages that were sent to the oracles $O_S$ and $O_V$, respectively, during an experiment.

A confirmer signature scheme must meet the following security requirements.

- *Unforgeability of signatures:* There exists no polynomial time algorithm which, on input $y_S$, $y_C$, $x_C$ (computed using the key generators) and given access to the oracle $O_S$, outputs with non-negligible probability an arbitrary correct message-signature pair $(m', \sigma')$ for a message $m' \notin M_S$.
- *Invisibility of signatures:* There exists a polynomial time algorithm *Sim* which on input $y_S$ and $y_C$ simulates signatures on arbitrary messages and has the following property. Let $B$ be a black-box which for fixed public keys and on input $m$ returns either simulated signatures (if $B = Sim$) or correct signatures (if $B = Sig$), and let $M_B$ denote the set of messages sent to $B$ during an experiment. Then for any polynomial time algorithm $A$ with access to $O_S$, $O_V$, and $B$, and which guarantees that $M_B$ and $M_S \cup M_V$ are always disjoint, the value

  $$|\Pr[A(y_S, y_C) = 1 \mid B = Sim] - \Pr[A(y_S, y_C) = 1 \mid B = Sig]|$$

  is negligible (the public keys are chosen according to the key generators). Note that the simulator *Sim* does not need the message $m$ as input. This makes the invisibility property even stronger than the usual *simulatability* property since the message of a correct message-signature pair $(m, \sigma)$ can be replaced by any message $m'$ (as long as $m'$ is chosen independently of $\sigma$) without enabling a verifier to find out that $(m', \sigma)$ is incorrect.
- *Consistency of verification:* For all $C^*_{Ver}$ and all (correct and incorrect) message-signature pairs $(m, \sigma)$ the following equation must hold

  $$\langle C^*_{Ver}(x_C), V_{Ver}() \rangle (m, \sigma, y_S, y_C) =_V \begin{cases} 1 & \text{if } (m, \sigma) \text{ is correct} \\ 0 & \text{otherwise} \end{cases}$$

  except with negligible probability. Informally, this means that the confirmer cannot claim that a correct (incorrect) signature is incorrect (correct).
- *Non-transferability of verification:* The verification protocol $\langle C_{Ver}, V_{Ver} \rangle$ is a minimum knowledge bi-proof (according to the definition given in [15]).

Note that these security requirements are very similar to the requirements for convertible undeniable signature schemes presented in [11].

## 3 A Weakness in a Previous Scheme

In this section we review one of the confirmer signature schemes proposed in [26] and point out that the confirmer can forge signatures of the signers. Let $G$ be a cyclic group of prime order $q$ and let $g$ be a generator of $G$. Let $h$ denote a collision resistant hash function. The scheme works as follows.

- *Key Generation:* The signer picks a random number $x_S$ as his secret key and computes $y_S := g^{x_S}$ as his public key. The confirmer picks a random number $x_C$ as his secret key and computes $y_C := g^{x_C}$ as his public key.
- *Signature Generation:* The signer computes $r_1 := g^{k_1}, c := y_C^{k_1}, r_2 := g^{k_2}$ using random $k_1, k_2 \in_R \mathbb{Z}_q$ and calculates $r := c \oplus h(m, r_2), s := k_2 + r \cdot x_S \pmod q$ (where $\oplus$ denotes bit-wise exclusive or). Then the triple $(r_1, r, s)$ is a designated confirmer signature on $m$.
- *Signature Verification:* Given a signature $(r_1, r, s)$ on a message $m$, the signer (or confirmer) and verifier compute $\tilde{c} := r \oplus h(m, g^s \cdot y_S^{-r})$. The confirmer gives an interactive zero-knowledge proof that $\log_{y_C} g \equiv \log_{r_1} \tilde{c} \pmod q$ holds or not by using the protocols in [4]. The signer can confirm a signature by giving an interactive zero-knowledge proof that $\log_{y_C} \tilde{c} \equiv \log_g r_1 \pmod q$ holds.

This scheme suffers from the weakness that the confirmer can forge signatures universally, i.e. he is able to generate signatures for all messages chosen by himself. He picks $r, s$ at random and computes for an arbitrary message $m$ the values $\tilde{c} := r \oplus h(m, g^s \cdot y_S^{-r})$ and $r_1 := \tilde{c}^{x_C^{-1}}$. The signature $(r_1, r, s)$ is valid, as $x_C \equiv \log_g y_C \equiv \log_{r_1} \tilde{c} \pmod q$. Furthermore, there is a $k_1$ such that $r_1 := g^{k_1}$ and thus $\tilde{c} = g^{k_1 \cdot x_C} = y_C^{k_1}$. Hence the equation $k_1 \equiv \log_g r_1 \equiv \log_{y_C} \tilde{c} \pmod q$ is satisfied as well.

The attack can also be adopted to the general construction based on proofs of knowledge and therefore also to special instances as the Fiat-Shamir based scheme [26]. If $r_1$ is an additional input of the hash function, the attack is countermeasured.

## 4 Confirmer Commitments

### 4.1 Definition

A confirmer commitment scheme (*CKG*, *Com*, *Sh*) consists of a commiter, a confirmer, a verifier, and the following efficient algorithms and protocols:

- A key generation algorithm $CKG(1^\ell) \rightarrow (x_C, y_C)$, where $\ell$ is a security parameter and $(x_C, y_C)$ is a secret/public key pair for the confirmer.
- A probabilistic algorithm $Com(m, y_C) \rightarrow d$ for committing to messages $m \in \{0,1\}^*$, where $y_C$ is the public key of the confirmer, and $d$ is the resulting commitment.

- An interactive showing protocol $Sh = \langle C_{Sh}, V_{Sh} \rangle$ between the confirmer and the verifier:

$$\langle C_{Sh}(x_C), V_{Sh}()\rangle(m, d, y_C) \rightarrow_V \begin{cases} 1 \\ 0 \end{cases} .$$

A commitment $d$ is called a commitment for a message $m$ (with respect to the confirmer public key $y_C$) if and only if $\Pr[Com(m, y_C) = d] > 0$. The above algorithms must meet the following requirements.

- *Collision resistance:* There exists no polynomial time algorithm which on input $x_C$ and $y_C$ outputs with non-negligible probability a triple $(m, m', d)$ such that $d$ is a commitment for both $m$ and $m'$.
- *Invisibility of commitments:* There exists a polynomial time simulator *CSim* such that for all messages $m$ the two random variables $Com(m, y_C)$ and $CSim(y_C)$ are computationally indistinguishable (according to the definitions of [18]).
- *Consistency of showing:* For all polynomial time algorithms $C^*_{Sh}$, for all messages $m$ and for all commitments $d$ the following equation must hold

$$\langle C^*_{Sh}(x_C), V_{Sh}()\rangle(m, d, y_C) =_V \begin{cases} 1 & \text{if } d \text{ is a commitment for } m \\ 0 & \text{otherwise} \end{cases}$$

  except with negligible probability.
- *Non-transferability of showing:* The protocol $\langle C_{Sh}, V_{Sh} \rangle$ is a minimum knowledge bi-proof (according to the definition given in [15]).

*Remark:* Note that the commiter is always able to prove that $d = Com(m, y_C)$ is a commitment for a message $m$ (but usually he is not able to prove that $d$ is not a commitment for $m' \neq m$).

Confirmer commitment schemes have many similarities with probabilistic encryption schemes [17]. In fact probabilistic encryption schemes are a special case of confirmer commitments with the additional property that the message $m$ can be derived (i.e. decrypted) from the commitment (cipher-text) $d$.

### 4.2 A Realization Based on the Decision-Diffie-Hellman Problem

Let $G$ be a cyclic group of prime order $q$ for which it is infeasible to compute discrete logarithms. Let further $h_q : \{0,1\}^* \rightarrow Z_q$ denote a collision resistant hash function. Then a secure confirmer commitment scheme can be constructed as follows.

- *Key generation:* $y_C = (g, z)$ for $g$ a randomly chosen generator of $G$, $z := g^{x_C}$, and $x_C \in_R \mathbb{Z}_q^*$
- *Commitment:* $Com(m, y_C = (g, z)) = (d_1, d_2)$ with $d_1 := g^{k+h_q(m)}$ and $d_2 := z^k$ for $k \in_R \mathbb{Z}_q$.

- *Showing:* $\langle C_{Sh}(x_C), V_{Sh}()\rangle(m, (d_1, d_2), y_C = (g, z))$ is an interactive zero-knowledge bi-proof for proving

$$\log_g(z) \overset{?}{\equiv} \log_{d_1}(d_2 z^{h_q(m)}) \pmod{q}.$$

An efficient protocol for such a zero-knowledge bi-proof can be found in [22] and is reviewed in the appendix B.

For the security analysis we make use of the following assumption.

**Decision-Diffie-Hellman Assumption:** Let the two sets $\mathcal{Q}$ and $\mathcal{DH}$ be defined as follows:

$$\mathcal{Q} \overset{\text{def}}{=} \{(g_1, g_2, y_1, y_2) \in G^4 \mid \langle g_1\rangle = \langle g_2\rangle = G\}$$
$$\mathcal{DH} \overset{\text{def}}{=} \{(g_1, g_2, y_1, y_2) \in \mathcal{Q} \mid \log_{g_1} y_1 = \log_{g_2} y_2\}$$

Note that the elements of $\mathcal{DH}$ correspond to a Diffie-Hellman key exchange with base base element $g_1$, exchanged values $y_1$ and $g_2$, and resulting key $y_2$. The DDH assumption says that two random variables which are uniformly distributed over $\mathcal{Q}$ and $\mathcal{DH}$, respectively, are computationally indistinguishable.

Let us now show briefly that the above scheme meets all security requirements.

- *Collision resistance:* Assume that there exists an algorithm for finding a collision, i.e. a triple $(m, m', d = (d_1, d_2))$ for which $d$ is a commitment for both $m$ and $m'$. This implies that there exists a $k \in \mathbb{Z}_q$ satisfying

$$z^k = d_2 \quad \text{and} \quad g^{k+h_q(m)} = d_1 = g^{k+h_q(m')} ,$$

which further implies that $h_q(m) = h_q(m')$. Therefore, the messages $m$ and $m'$ are a collision for the hash function $h_q$ which contradicts the assumptions.

- *Invisibility:* The simulator for this commitment scheme consists of an algorithm that outputs a pair of truly random elements of $G$. For any message $m$ it is possible to show that the invisibility property of the commitment scheme is equivalent to the DDH assumption:

  - *Invisibility $\rightarrow$ DDH:* a single sample $(d_1, d_2)$ of the invisibility problem with public key $y_C = (g, z)$ can be transformed into arbitrary many samples of the DDH problem by computing

$$(g^\alpha, z^\beta, d_1^{\alpha\gamma} g^{\alpha\delta}, (d_2 z^{h_q(m)})^{\beta\gamma} z^{\beta\delta})$$

  for $\alpha, \beta \in_R \mathbb{Z}_q^*$ and $\gamma, \delta \in_R \mathbb{Z}_q$. If $(d_1, d_2)$ is a commitment for $m$, then the resulting DDH samples are uniformly distributed over $\mathcal{DH}$. If $(d_1, d_2)$ is simulated, then the DDH samples are uniformly distributed over $\mathcal{Q}$ (except with negligible probability).

  - *DDH $\rightarrow$ invisibility:* a single sample $(g_1, g_2, y_1, y_2)$ of the DDH problem can be transformed into arbitrary many samples of the invisibility problem by computing

$$(g = g_1,\ z = g_2, d_1 = y_1^{\alpha} g_1^{\beta + h_q(m)}, d_2 = y_2^{\alpha} g_2^{\beta})$$

for $\alpha, \beta \in_R \mathbb{Z}_q$. If the DDH sample was in $\mathcal{DH}$, the resulting samples are random commitments for $m$ with regard to the public key $y_C = (g, z)$. Otherwise, the resulting samples are distributed according to the distribution of the simulator.

- *Consistency and non-transferability of showing:* These two properties are direct consequences of the soundness property and the zero-knowledge property, respectively, of the interactive bi-proof system. The proofs of these two properties can be found in [22].

## 5 Confirmer Signatures Using Confirmer Commitments

At a first glance, an obvious realization of a confirmer signature scheme based on a confirmer commitment scheme and any secure digital signature scheme would be the following. Instead of signing a message $m$ directly, i.e. computing $\mathit{Sig}(m)$, the signer first computes a confirmer commitment $d = \mathit{Com}(m, y_C)$ and then generates a signature on this commitment, i.e. $\sigma = \mathit{Sig}(d)$. A verifier can check that $\sigma$ is a correct signature on $d$, but since he cannot check whether $d$ is a commitment for $m$, the verification is only possible with the cooperation of the confirmer. Such a scheme is sufficient in many practical applications, but it does not satisfy our security requirements. More presicely, it does not meet the requirements for the invisibility property because in general no simulation of signatures is possible.

In this section we show that for two large classes of digital signature schemes, namely those based on 3-move proofs of knowledge and existentially forgeable signatures, it is possible to design related confirmer signature schemes that meet the requirements defined in Section 2.

### 5.1 Proofs of Knowledge and Proof-Based Signatures

Let $\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$ be a polynomial time checkable relation for which random samples $(x, y) \in \mathcal{R}$ can be computed efficiently, but for which it is infeasible, given $y \in \mathcal{Y}$, to find an $x \in \mathcal{X}$ such that $(x, y) \in \mathcal{R}$. Let $\langle P, V \rangle$ be a proof of knowledge for the relation $\mathcal{R}$ (see [24] for definitions) between a prover $P$ and a verifier $V$. Informally, on common input $y$ the protocol $\langle P, V \rangle$ allows $P$ to convince $V$ about his knowledge of a value $x$ satisfying $(x, y) \in \mathcal{R}$. We assume that $\langle P, V \rangle$ has the following properties:

- $\langle P, V \rangle$ consists of exactly three moves and the only active role of the verifier $V$ is to announce a random $t$-bit challenge in the second move. The value $t$ is a security parameter and is chosen such that successful cheating is possible only with negligible probability (e.g. $t = O(|y|)$).

$$\begin{array}{lcl}
\underline{P(x,y)} & & \underline{V(y)} \\
(r,k) := P_1(x,y) & \xrightarrow{\quad r \quad} & \\
 & & c \in_R \{0,1\}^t \\
 & \xleftarrow{\quad c \quad} & \\
s := P_3(x,y,r,k,c) & & \\
 & \xrightarrow{\quad s \quad} & Accept(r,c,s,y) \rightarrow \begin{cases} 1 \\ 0 \end{cases}
\end{array}$$

After the last move, the verifier decides whether to accept the proof or not.

- $\langle P, V\rangle$ is honest-verifier zero-knowledge in the following way: there exists a simulator $Sim_{\langle P,V\rangle}$ which on input $y$ and a challenge $c$ outputs a triple $(r', c' = c, s')$ such that for a randomly chosen $t$ bit string $c$ the random variable $Sim_{\langle P,V\rangle}(y, c)$ is indistinguishable from a transcript of a successful execution of $\langle P, V\rangle$ on input $y$ (note that $V$ is the honest verifier).

Efficient proof systems of this type are known for several relations, such as discrete logarithms in a finite cyclic group $G = \langle g\rangle$ (with $\mathcal{R}_{DL} = \{(x, g^x) \mid 0 \leq x < |G|\}$) [31], and quadratic residues modulo a composite number $n$ (with $\mathcal{R}_{QR} = \{(u, u^2 \bmod n) \mid u \in \mathbb{Z}_n^*\}$) [14].

Given a collision resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^t$, such a proof system can be used to build a digital signature scheme (with the techniques presented in [24]). The signer chooses a random secret/public key pair $(x, y) \in_R \mathcal{R}$. A signature on a message $m$ is generated as follows: $(r, k) := P_1(x, y)$; $c := h(m, r)$; $s := P_3(x, y, r, k, c)$. The resulting signature is the pair $(r, s)$ and can be verified by checking whether $Accept(r, h(m, r), s, y) = 1$.

Using the techniques of [2, 29] it can be proved that under the assumption that $h$ is a truly random function, breaking such a signature scheme (with an adaptive chosen message attack) is as hard as finding the secret key $x$.

## 5.2 Proof-based Confirmer Signatures

Using a confirmer commitment scheme a proof of knowledge of the type described above can also be turned into a secure confirmer signature scheme in the following way ($h$ is the collision resistant hash function used above).

- *Key generation:* $(x_S, y_S) \in_R \mathcal{R}$ and $(x_C, y_C) := CKG(1^\ell)$
- *Signature generation:*
  - $(r, k) := P_1(x_S, y_S)$
  - $d := Com(m\|r, y_C)$, where $\|$ denotes concatenation of strings
  - $c := h(d)$
  - $s := P_3(x_S, y_S, r, k, c)$

  The resulting signature is $(r, s, d)$.
- *Signature verification:* The verifier's output is 0 if $Accept(r, h(d), s, y_S) = 0$, and $\langle C_{Sh}(x_C), V_{Sh}()\rangle(m\|r, d, y_C)$ otherwise.

In other words, a message-signature pair $(m,(r,d,s))$ is regarded correct if and only if $\mathit{Accept}(r,h(d),s,y_S)=1$ and $d$ is a commitment for $m\|r$.

Under the assumption that $h$ is a truly random function it can be shown that this scheme meets all security requirements of a confirmer signature scheme.

- *Unforgeability of signatures:* We show that any algorithm $A$ that can forge signatures with non-negligible probability can be used to find the signer's secret $x_S$ which contradicts the assumptions about the relation $\mathcal{R}$. Without loss of generality we assume that $A$ evaluates $h$ for distinct arguments only (see also [29]). We take control over $h$ to simulate queries to the oracle $O_S$ (this simulation is not done by using the simulator of the confirmer signature scheme but by taking control over $h$): on input $m$ we first compute $(r,c,s):=\mathit{Sim}_{\langle P,V\rangle}(y_S,c)$ for a random $c$, then we compute $d:=\mathit{Com}(m\|r,y_C)$, set $h(d):=c$, and return the signature $(r,s,d)$ as the result of the simulated oracle. Since $h$ is still a random function, $A$ will finally return a correct signature $(r',s',d')$ on a message $m$ with the same probability of success. Using the proof techniques of [29] we can guess with sufficiently high probability the position where $h$ is evaluated with $d'$ and return a new random value. With non-negligible probability $A$ outputs another signature $(r',s'',d')$ (note that $r'$ remains fixed because the confirmer commitment scheme is collision resistant). This method can be repeated until there are enough $s$ values such that the secret key $x_S$ can be computed with the knowledge extractor of the proof system.
- *Invisibility of signatures:* The simulator for this signature scheme works as follows: using the simulator *CSim* of the commitment scheme, one computes $d':=\mathit{CSim}(y_C)$ and $c':=h(d')$. Then, using the simulator of the proof system, one computes $(r',c',s'):=\mathit{Sim}_{\langle P,V\rangle}(y_S,c')$; the resulting simulated signature on $m$ is $(r',s',d')$. We show that with this simulator, the invisibility property of the signature scheme is equivalent to the invisibility property of the confirmer commitment scheme. As above, we take control over the function $h$ and simulate queries to $O_S$. Queries to the oracle $O_V$ can be simulated as follows: on input $(m,(r,s,d))$ we return 1 if and only if $\mathit{Accept}(r,h(d),s,y_S)=1$ and $O_S$ has returned a signature containing $r$ and $d$ (note that in all other cases the signature must be incorrect because of the unforgeability property). Now it can easily be seen that if the original algorithm was able to distinguish correct and simulated signatures, the modified algorithm would be able to distinguish real commitments from simulated ones.
- *Consistency and non-transferability of verification:* These two properties are direct consequences of the corresponding properties of the confirmer commitment scheme.

**Example:** Schnorr's identification scheme [31] is a typical example of a proof of knowledge that can be converted into a confirmer signature scheme. Let $q$ and $p$ be prime numbers with $q\approx 2^{200}$, $p\approx 2^{1024}$, and $q|(p-1)$. Let further $g$

denote a generator of the subgroup $G$ of $\mathbb{Z}_p^*$ of order $q$. The signer's public key is $y_S = g^{x_S}$ for a randomly chosen $x_S \in_R \mathbb{Z}_q$. A message $m$ is signed by computing $r := g^k$ for $k \in_R \mathbb{Z}_q$, $d := Com(m\|r, y_C)$, and $s := k - h(d)x_S \pmod q$. The resulting signature is the pair $(s, d)$. Note that in the Schnorr scheme $r$ equals $g^s y_S^{h(d)}$ and therefore $r$ need not be part of the signature. The signature can be verified by asking the confirmer to show that $d$ is a commitment for $m\|r$.

With the confirmer commitment scheme described in section 4.2 (using the same group $G$) the efficiency of this Schnorr-based confirmer scheme is as follows. The signature is 2248 bits long and its generation requires roughly 900 multiplications modulo $p$. During the interactive verification of a signature, 6120 bits of data are exchanged, the confirmer computes about 2250 modular multiplications, and the verifier computes about 2450 modular multiplications (see appendix B). This compares favourably with the schemes presented in [11, 22].

## 5.3 Existentially Forgeable Signatures

Apart from the proof-based signature schemes, there exists also a generic construction for existentially forgeable signatures. More precisely, we consider signature schemes $(KG^*, Sig^*, Ver^*)$ that meet the following requirements.

- $KG^*$ is a probabilistic key generation algorithm which outputs a secret/public key pair $(x_S, y_S)$, $Sig^*$ is the signature generation function which on input a message $m$ and the signer's secret key $x_S$ outputs a signature $\sigma^*$, and $Ver^*$ is a verification function which on input $m$, $\sigma^*$, and the signer's public key $y_S$ outputs 1 if $\sigma^*$ is a correct signature on $m$ and 0 otherwise.
- Let $\mathcal{M}$ be the (finite) set of all messages that can be signed. There exists a polynomial time algorithm *ExistForge* which on input $y_S$ outputs a message-signature pair $(m, \sigma)$. Let $M\Sigma$ denote the random variable representing the output of *ExistForge* and let $P_{M\Sigma}(m, \sigma)$ denote the probability that $M = m$ and $\Sigma = \sigma$. Then for all $m \in \mathcal{M}$ with $P_M(m) > 0$ the two distributions $P_{\Sigma|M}(\sigma, m)$ and $\Pr[Sig(m, x_S) = \sigma]$ over all possible values of $\sigma$ are indistinguishable. In other words, if messages are only chosen according to the probability distribution of $M$ then real and forged message-signature pairs are indistinguishable.
- There exists a finite group $B$ with binary operation $\odot$ and inversion function *inv* (both efficiently computable) such that the probability distribution of messages generated by *ExistForge* and the uniform distribution over $B$ are indistinguishable.
- Let $h_B : \{0,1\}^* \to B$ be a collision resistant hash function. Then signatures of the form $Sig(h_B(m))$ are existentially unforgeable under adaptive chosen message attacks.

To illustrate these requirements let us consider the RSA scheme [30] with composite modulus $n$, public exponent $e$, and $\mathcal{M} = \mathbb{Z}_n^*$. A simple method for existentially forging RSA signatures is the following: first, choose $\sigma \in \mathbb{Z}_n^*$ at random and then compute the message $m := \sigma^e \pmod{n}$. In this case, the random variable $M$ is uniformly distributed over $\mathcal{M}$. Hence, we can for instance define $B = \mathbb{Z}_n^+$, i.e. the additive group of integers modulo $n$ (note that the uniform distributions over $\mathbb{Z}_n^*$ and $\mathbb{Z}_n^+$, respectively, are indistinguishable).

It is easy to see that the ElGamal signature scheme [13] and some of its variants satisfy these conditions as well.

### 5.4 Confirmer Signatures Based on Existentially Forgeable Signatures

Let $(KG^*, Sig^*, Ver^*)$ be an existentially forgeable signature scheme of the type defined above. Using a confirmer commitment scheme $(CKG, Com, Sh)$ it is possible to construct a confirmer signature scheme as follows.

- *Key generation:* $KG_S := KG^*$ and $KG_C := CKG$.
- *Signature generation:*
  1. $b \in_R B$
  2. $d := Com(m \| b, y_C)$
  3. $\sigma^* := Sig^*(h_B(d) \odot b, x_S)$

  The resulting signature is the triple $(\sigma^*, d, b)$.
- *Signature verification:* The verifier's output is 0 if $Ver^*(h_B(d) \odot b, \sigma^*, y_S) = 0$, and $\langle C_{Sh}(x_C), V_{Sh}()\rangle(m\|b, d, y_C)$ otherwise.

The security proof is sketched in the appendix A. With the RSA example given above and any confirmer commitment scheme, the signature is computed with $b \in_R \mathbb{Z}_n^+$, $d := Com(m\|b, y_C)$, and $\sigma^* := (h_B(d) + b)^{1/e} \pmod{n}$. The efficiency of this scheme with small public exponent $e$ compares favourably with the undeniable signature scheme of [16].

## 6 Extensions

In this section we suggest another realization of the confirmer commitment based on an alternative number theoretic assumption. Then it is outlined how to deal with multiple confirmers. Moreover, based on any efficient confirmer signature scheme an efficient fair contract signing scheme with off-line trusted party (see e.g. [1] for the model) can be realized, but details are omitted here.

### 6.1 Confirmer Commitment Scheme Based on the Higher Residues

It is desirable to design confirmer commitments based on other well established number theoretic assumptions. One candidate is the $e$-th residue assumption (e.g. see [9, 25]): An element $a$ is an *e-th residue* in $G$, if and only if there is an element $w \in G$ such that $w^e = a$ (let denote this set by $HR_e$). All elements in $G$ which are not $e$-th residues are called $e$-th *non-residues.* Loosely speaking, it is assumed that for certain groups it is hard to decide (better than guessing) if a given number $w \in G$ is an $e$-th residue, i.e. whether $w \in HR_e$. For example, $G = \mathbb{Z}_n^*$, $n$ is composite with unknown factorization and $e$ (but not $e^2$) divides $\varphi(n)$. Using the techniques of [9, 25] combined with the proof technique of [19, 23] it is possible to construct a confirmer commitment scheme. However, this solution is unsatisfactory since there is no efficient method for proving that a given $a \in G$ is an element of $G \backslash HR_e$. By modifying the confirmer commitment model and the general constructions for confirmer signatures slightly, we can get around this problem. The basic idea is to fix a *taboo-message* $m'$ which must not be committed in the application. A commitment is *well-formed* if and only if it is either a commitment of a certain message $m \neq m'$ or of the taboo-message $m'$. Simulated commitments are just commitments of the taboo message. In the interactive showing protocol the confirmer either proves that a given commitment is a commitment of the actual message (which results in output 1) or of the taboo message (resulting in output 0). To do this, only an efficient proof for $w \in HR_e$ is required. However, this works only if everybody is able to check that a commitment is well-formed. Therefore, a minimum knowledge argument is added to each commitment in order to prove that the commitment is well formed. Such proofs can easily be constructed using the techniques of [10] for proving that one of two statements is true without revealing which.

## 6.2 Multiple Confirmers

An obvious extension of the basic confirmer signature model is to allow several confirmers $C_1, \ldots, C_n$ to verify signatures.

For instance, this can be achieved by modifying the confirmer commitment scheme as follows: a multi-confirmer commitment for a message $m$ is a $n$-tuple of individual commitments, i.e. $(Com(m, y_{C_1}), \ldots, Com(m, y_{C_n}))$. A verifier now needs to ask only a single confirmer to verify a signature. However, in order to guarantee the consistency of verification, the signer has to provide a proof that the individual commitments in the tuple are all commitments of a single message. For the discrete logarithm based solution presented in Section 4 this is very simple: for the $n+1$-tuple $(g^{\alpha + h_q(m)}, y_{C_1}^{\alpha}, \ldots, y_{C_n}^{\alpha})$ the signer proves that discrete logarithms of the last $n$ group elements to the corresponding bases are all equal. It is possible that the signer is one of the confirmers. This enables him to confirm correct signatures and/or to deny incorrect signatures on his own. More complex access structures are possible as well, e.g. a scheme where the collaboration of at least $k$ out of the $n$ confirmers is required to verify a signature.

## 7 Acknowledgements

The authors would like to thank the anonymous referees for very useful comments. This research was supported by the Swiss National Foundation (SNF/SPP project no. 5003-045293).

## References

1. N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures", Research Report RZ 2973 (#93019), IBM Research, November, (1997).
2. J. Bellare, P. Rogaway, "Random Oracles are practical: a paradigm for designing efficient protocols", Proc. 1st ACM Conference on Computer and Communications Security, ACM Press, (1993), pp. 62–73.
3. J. Boyar, D. Chaum, I. Damgard, T. Pedersen, "Convertible undeniable signatures", LNCS 537, Proc. Crypto '90, Springer Verlag, (1991), pp. 189–205.
4. D. Chaum, "Zero-knowledge undeniable signatures", LNCS 473, Proc. Eurocrypt '90, Springer Verlag, (1991), pp. 458–464.
5. D. Chaum, "Some weakness of "Weaknesses of Undeniable Signatures"", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 554–556.
6. D. Chaum, "Designated confirmer signatures", LNCS 950, Proc. Eurocrypt '94, Springer Verlag, (1995), pp. 86–91.
7. D. Chaum, H. van Antwerpen, "Undeniable Signatures", LNCS 435, Proc. Crypto '89, Springer Verlag, (1990), pp. 212–216.
8. D. Chaum, T. Pedersen, "Wallet databases with observers", LNCS 740, Proc. Crypto'92, Springer Verlag, (1993), pp. 89 - 105.
9. J.D. Cohen, M.J. Fischer, "A robust and verifiable cryptographically secure election scheme", Proc. 26th FOCS, (1985), pp. 372–382.
10. R. Cramer, I. Damgård, B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols", LNCS 839, Proc. Crypto'94, Springer Verlag, (1994), pp. 174–87.
11. I. Damgård, T. Pedersen, "New convertible undeniable signature schemes", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 372–386.
12. Y. Desmedt, M. Yung, "Weaknesses of undeniable signature schemes", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 205–220.
13. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, Vol. IT-30, No. 4, July, (1985), pp. 469 - 472.
14. A. Fiat, A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", LNCS 263, Proc. Crypto '86, Springer Verlag, (1987), pp. 186–194.
15. 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.
16. R. Gennaro, H. Krawczyk, T. Rabin, "RSA-based undeniable signatures", LNCS 1294, Proc. Crypto'97, Springer Verlag, (1997), pp. 132–149.
17. S. Goldwasser, S. Micali, "Probabilistic Encryption", Journal of Computer and System Sciences, vol. 28, no. 2, (1984), pp. 270–299.
18. S. Goldwasser, S. Micali, C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", SIAM Journal on Computing, vol. 18, no. 1, (1989), pp. 186–208.
19. L.C. Guillou, J.-J. Quisquater, "A paradoxical identity based signature scheme resulting from zero-knowledge", LNCS 403, Proc. Crypto'88, Springer Verlag, (1989), pp. 465–473.
20. M. Jakobsson, "Blackmailing using undeniable signatures", LNCS 950, Proc. Eurocrypt'94, Springer Verlag, (1995), pp. 425–427.
21. 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.

22. M. Michels, M. Stadler, "Efficient convertible undeniable signature schemes", Proc. 4th Annual Workshop on Selected Areas in Cryptography (SAC'97), (1997), pp. 231–243.
23. K. Ohta, T. Okamoto, "A modification of the Fiat-Shamir scheme", LNCS 403, Crypto'88, (1989), pp. 232–244.
24. J. Feige, A. Fiat, A. Shamir, "Zero-Knowledge proofs of identity", Journal of Cryptology, Vol. 1, No. 1, (1988), pp. 77–94.
25. K. Kurusawa, K. Katayama, W. Ogata, S. Tsujii, "General public key residue cryptosystem and mental poker protocols, LNCS 473, Proc. Eurocrypt'90, Springer Verlag, (1991), pp. 374–387.
26. T. Okamoto, "Designated confirmer signatures and public-key encryption are equivalent", LNCS 839, Proc. Crypto'94, Springer Verlag, (1994), pp. 61–74.
27. T.P.Pedersen, "Distributed provers with applications to undeniable signatures", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 221–242.
28. S. J. Park, K. H. Lee, D. H. Won, "An entrusted undeniable signature scheme", Proc. Japan-Korea Workshop on Information Security and Cryptography, (1995), pp. 120–126.
29. D. Pointcheval, J. Stern, "Security proofs for signature", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 387–398.
30. R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", CACM, vol. 21, no. 2, (1978), pp. 120–126.
31. C. P. Schnorr, "Efficient signature generation for smart cards", Journal of Cryptology, Vol. 4, (1991), pp. 161–174.

## A Sketch of a Security Proof

Under the assumption that $h_B$ is a truly random functions it can be shown that this scheme meets all security requirements of a confirmer signature scheme.

- *Unforgeability of signatures:* Let $h'$ denote another truly random function. Using the techniques of [29] we show that forging confirmer signatures is as hard as forging digital signatures of the underlying scheme using the hash function $h'$. Let $A$ be an efficient algorithm for forging signatures. We take control over $h$ to simulate the oracle $O_S$ (see the proof of the proof-based signatures). With non-negligible probability $A$ will return a correct message-signature pair $(m, (\sigma^*, d, b))$. We guess the position at which $h_B$ is evaluated with the argument $d$ and replace its output by $h'(m) \odot \mathit{inv}(b)$. Again with non-negligible probability, $A$ will return a correct message-signature pair $(m, (\sigma', d, b'))$. Since the confirmer commitment scheme is assumed to be collision resistant, $b$ and $b'$ must be equal. But then we have $h_B(d) \odot b' = h'(m)$ and therefore $\sigma'$ is a correct signature on $m$ in the secure variant of the underlying signature scheme using the hash function $h'$.
- *Invisibility of signatures:* A simulated signature for a message $m$ can be computed as follows: using the simulators of the commitment scheme and of the existentially forgeable signature scheme one computes $d := \mathit{CSim}(y_C)$ and $(m^*, \sigma^*) := \mathit{ExistForge}(y_S)$, respectively. Then, one computes $b := \mathit{inv}(h_B(d)) \odot m^*$ (note that $m^* \in B$ with overwhelming probability). The

resulting simulated signature is the triple $(\sigma^*, d, b)$. Under the assumption that $h_B$ is a truly random function, on can show that simulated and "real" signatures can be distinguished only if the invisibility property of the confirmer commitment scheme is violated.

- *Consistency and non-transferability of verification:* These two properties are direct consequences of the corresponding properties of the confirmer commitment scheme.

## B A Bi-Proof for the Equality of two Discrete Logarithms

An important component of the confirmer commitment scheme based on the DDH assumption 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. An algorithm for proving the equality and another one for proving the inequality was first given by Chaum in [4]. Both algorithm can also be combined into a bi-proof. A bit-wise bi-proof for this statement can be found in [15]. Because of efficiency reasons we review the protocol given in [22] here.

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 $\beta$ and $z$ in a multiplicative group $G$ of prime order $q$, in which computing discrete logarithms is infeasible.

1. The verifier chooses random values $u, v \in \mathbb{Z}_q$, computes $a := \alpha^u y^v$, and sends $a$ to the prover.
2. The prover chooses random values $k, \tilde{k}, w \in \mathbb{Z}_q$, computes $r_\alpha := \alpha^k$, $r_\beta := \beta^k$, $\tilde{r}_\alpha := \alpha^{\tilde{k}}$, and $\tilde{r}_\beta := \beta^{\tilde{k}}$, and sends $r_\alpha$, $r_\beta$, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, 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 $s := k - (v+w)x (\text{mod } q)$, $\tilde{s} := \tilde{k} - (v + w)k (\text{mod } q)$ 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:

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

In [22] it was proved that the above protocol is complete and sound. It is also 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.

Note that if just the equality of two discrete logarithms should be proved the protocol can be simplified such that it equals Chaum's equality protocol in [4]. 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 $v = \mathcal{H}(\alpha, y, \beta, z, r_\alpha, r_\beta, \tilde{r}_\alpha, \tilde{r}_\beta)$, where $\mathcal{H}$ is a collision resistant hash function.