New Convertible Undeniable Signature Schemes

Ivan Damgård
Aarhus University, Computer Science Department, BRICS,
Ny Munkegade, DK-8000 Århus C
and
Torben Pedersen
Cryptomathic,
Århus Science Park, Gustav Wieds Vej 10, DK-8000 Århus C

Abstract. Undeniable signatures are like ordinary digital signatures, except that testing validity of a signature requires interaction with the signer. This gives the signer additional control over who will benefit from being convinced by a signature, and is particularly relevant when signing sensitive, non-public data. Convertible undeniable signatures offer additional flexibility in that there is a separate verification key that can be used to verify a signature (without interaction). This allows the signer to delegate the ability to verify signatures to one or more participants, and ultimately to convert all signatures to ordinary ones by making the verification key public. While provably secure theoretical solutions exist for convertible schemes, earlier practical schemes proposed have either been broken or their status as far as security is concerned is very unclear. In this paper, we present two new convertible schemes, in which forging signatures is provably equivalent to forging El Gamal signatures. The difficulty of verifying signatures without interacting with the signer is based on the factoring problem for one of the schemes and on the Diffie-Hellman problem for the other scheme.

1 Introduction and Related Work

Undeniable signatures are like ordinary digital signatures, except that testing validity of a signature requires interaction with the signer. There must be an interactive protocol, both for verifying and disavowing a signature. This gives the signer additional control over who will benefit from being convinced by a signature, and is particularly relevant when signing sensitive, non-public data: Two parties entering into a confidential business agreement will of course want each other to be committed to the deal, but will certainly not want the contents of the agreement to become public, together with a signature anyone can verify. Undeniable signatures are clearly more suitable for this situation than classical ones, since the signer then has the option of saying "no comments" if the data and signature is published e.g. by the press. But note that in case of a legal dispute, he can still be required to confirm or deny the signature and could be considered bound to the signature if he refuses to cooperate.

Undeniable signatures were first introduced by Chaum and van Antwerpen [CvA90], who proposed a scheme based on discrete logarithms. The existence of undeniable signatures was proved to be equivalent to existence of one-way functions by Micali [Mic90].

In [DY91], Desmedt and Yung point out that a group of mutually distrusting verifiers could all get convinced about a signature while only executing one verification protocol with the signer. In general this requires that the verifiers do a multiparty computation. Since such computations are often feasibly only in theory, this may not be a very realistic attack. Nevertheless, it can be prevented completely using a technique known as designated verifier protocols suggested by Jakobson et al.[JKR96] (based partly on ideas by Chaum). In such a protocol, only the verfier whose public key is used in the proof will be convinced. The protocols we present in this paper can all be turned into designated verifier protocols.

Convertible undeniable signatures offer additional flexibility in that there is a separate verification key that can be used to verify a signature (without interaction). This allows the signer to delegate the ability to verify signatures to one or more participants, and ultimately to convert all signatures to ordinary ones by making the verification key public. As an application of this, consider the problem of keeping digital records of confidential political decisions. Authenticating such records with standard signatures is hardly acceptable: if the data leak to the press, anyone can verify the signatures. Undeniable signatures are clearly more suitable in this respect. However, such records usually become publicly accessible after some years, and should therefore also become publicly *verifiable*. However, the signer who generated the undeniable signatures may at this point be unable to handle the verification requests that may now be submitted to him: it may be infeasible because of the number of requests to handle, or the signer may not even be present. This can be solved using a convertible scheme: the signer could make the verification key public after a certain period, or give it initially to a trusted third party, who would release it later.

Even if signatures are never converted in the above sense, convertible schemes can still be useful: in many applications, signatures are generated once, but verified many times. If there is a separate verification key, it can be distributed to a large number of locations. This facilitates handling many verifications without compromising security of the secret key needed to generate signatures.

Convertible undeniable signatures were introduced by Boyar, Chaum, Damgård and Pedersen [BCDP91], who proved that such schemes exist if and only if one-way functions exist. They also proposed a practical scheme based on the discrete logarithm problem. This scheme, however, has recently been broken by Markus Michels [Mic95]. He also proposed a modification that seems secure against this attack, but the modified scheme does not seem to have a provable relation to any well established intractability assumption.

In this paper, we present two new convertible schemes, in which forging signatures is provably equivalent to forging El Gamal signatures. The difficulty of verifying signatures without interacting with the signer is based on the factoring problem for one of the schemes and on the Diffie-Hellman problem for the other scheme. The schemes are provably secure under appropriate intractability assumptions for the underlying signature and encryption schemes.

2 Definitions and Notation

In this section we define the concept of a (convertible) undeniable signature scheme. We use the standard concepts of interactive Turing machines, interactive proof systems and zero-knowledge without further explanation. The reader is referred to [GMR89] for details.

An undeniable signature scheme consists of the following 6 components:

- A *Key generator algorithm* GEN. This is a probabilistic polynomial time algorithm, which receives 1^k as input, where k is a security parameter, and generates as output a triple of keys (K_S, K_V, K_P), called the *secret, verification* and *public* key, respectively. The secret key K_S is used by the signer to create undeniable signatures, while the keys K_V and K_P are used by the signer and verifier, respectively, in the confirmation and disavowal protocols. The scheme defines a set of *legal public keys* called $\mathcal{L}$. Any K_P generated by GEN must be in $\mathcal{L}$.
  For some schemes, it will be the case that $K_V = K_S$. For others, including the convertible ones, they will be different. To apply the scheme, the signer will run GEN and publish K_P, while keeping K_S and K_V for himself. For convertible schemes, he may publish K_V at some later time, or distribute it to a limited set of parties.
- A *Signature algorithm* SIGN. This is a probabilistic polynomial time algorithm, which receives a secret key K_S and a message m, and outputs a signature s. The message, resp. the signature are in $\mathcal{M}$ resp. $\mathcal{S}$, which are sets of binary strings called the *message space* resp. the *signature space*. We note that since SIGN is allowed to be probabilistic, s may not be uniquely determined from m and K_S.
- A mapping VALID which given a public key $K_P \in \mathcal{L}$ and a message $m \in \mathcal{M}$ uniquely identifies a subset of the possible signatures. The intuition is that VALID(m, K_P) is the set of signatures valid w.r.t. a given public key and message.
- A *Verification protocol* (C, V_C). This is a pair of interactive polynomial time Turing machines called the *Confirmer* and the *Verifier.* The common input consists of a message $m \in \mathcal{M}$, a string $z \in \mathcal{S}$, and a public key K_P. The confirmer receives as private input a verification key K_V. Intuitively, z is a signature, which the confirmer claims is valid w.r.t. m and K_P. The protocol is designed to convince V_C about this.
- A *Disavowal protocol* (D, V_D). This is a pair of interactive polynomial time Turing machines called the *Denier* and the *Verifier.* The common input consists of a message $m \in M$, a string $z \in Z$ and public key K_P. The denier receives as private input a verification key K_V. Intuitively, z is a signature,

which the denier claims is invalid w.r.t. m and K_P. The protocol is designed to convince V_D about this.

- A *Signature simulator* SIGN_{Sim}. This is a probabilistic polynomial time algorithm which receives a message m and a public key K_P as input and outputs an element in S called a *simulated signature*. The intuition is that a simulated signature should look like a real signature to anyone who knows only public information. Therefore someone who receives a message and a purported signature from an untrusted source cannot tell on his own if the signature is valid, since it might as well be a simulated one.

In order for the scheme to make sense, the following basic properties are required from its components:

- The signature algorithm always produces valid signatures. More formally: Let the triple (K_S, K_V, K_P) be a possible output from GEN. Then, on input (m, K_S), SIGN always produces an output in $\text{VALID}(m, K_P)$.
- The signer can always confirm a correct signature without revealing any side information, but cannot convince the verifier that an incorrect signature is valid. As a part of this proof, he may need to also convince the verifier that the public key was correctly generated. More formally:
  The confirmation protocol is a zero-knowledge interactive proof system for the language $\{(m, z, K_P) | \; K_P \in \mathcal{L} \text{ and } z \in \text{VALID}(m, K_P)\}$, where we require completeness with probability 1.
- The signer can always disavow an invalid signature (no matter how it was produced) without revealing any side information, but cannot convince the verifier that a valid signature is incorrect. As a part of this proof, he may need to also convince the verifier that the public key was correctly generated. More formally:
  The disavowal protocol is a zero-knowledge interactive proof system for the language $\{(m, z, K_P) | \; K_P \in \mathcal{L} \text{ and } z \notin \text{VALID}(m, K_P)\}$, where we require completeness with probability 1.

We have required that both protocols, in addition to the statement on z, convince the verifier that $K_P \in \mathcal{L}$. Formally, this is necessary since $\text{VALID}(m, K_P)$ is undefined if $K_P \notin \mathcal{L}$. There may be a very real problem behind this, since for some schemes the signer could cheat in the verification or disavowal if $K_P \notin \mathcal{L}$.

For some schemes, including the ones we present here, $\mathcal{L}$ is polynomial time recognisable, in which case there is nothing extra to prove, the verifier can check himself that $K_P \in \mathcal{L}$.

In the following, an *undeniable signature scheme* should be taken to mean a 6-tuple $(\text{GEN}, \text{SIGN}, (C, V_C), (D, V_D), \text{VALID}, \text{SIGN}_{\text{Sim}})$ with the above description and properties. So far, we have only covered part of the security we want (by requiring (C, V_C) and (D, V_D) to be zero-knowledge interactive proofs). The rest of the security comes in two parts, one dealing with security against verifying signatures without knowing, K_V, and one dealing with security against forgeries.

For the first part we need to introduce a *distinguisher enemy* which is trying to verify a signature.

Definition 1. A *distinguisher enemy* E_D is a probabilistic polynomial time algorithm, which can be used in the following type of experiment:

1. GEN is executed on input 1^k, let the output be K_S, K_V, K_P. As input, E_D gets K_P and 1^k.
2. E_D may now make any number of *status requests* and *signature requests*. In a status request, E_D produces a pair (m, z), and receives a 1-bit answer which is 1, iff $z \in \text{VALID}(m, K_P)$. In a signature request, E_D produces a message m and receives the result of running SIGN on input m, K_S.
3. Let M be the set of messages occurring in status or signature requests done in step 2. Now E_D outputs a message $m_0 \notin M$, and receives a string z_0, which is either the result of running SIGN on m, K_S, or the result of running SIGN_{Sim} on m, K_P. We refer to these two cases as *the real case* and *the simulated case*, resp.
4. E_D may now make any number of status or signature requests, provided that m_0 does not occur as the message in any request, and z_0 does not occur in any status request.
5. Finally E_D outputs 1 bit.

We must now define what it means that E_D is capable of distinguishing the simulated and the real case:

Definition 2. Let $p_{\text{real}}(k)$, resp. $p_{\text{sim}}(k)$ be the probability that E_D outputs 1 in the real, resp. the simulated case above. These probabilities are taken over the random choices made by E_D, GEN and SIGN.

E_D is *successful* against the scheme defined by the 6-tuple

$$(\text{GEN}, \text{SIGN}, (C, V_C), (D, V_D), \text{VALID}, \text{SIGN}_{\text{Sim}}),$$

if there is a polynomial P such that for infinitely many k,

$$|p_{\text{real}}(k) - p_{\text{sim}}(k)| \geq 1/P(k).$$

The reader may notice that the verify and disavowal protocols do not enter explicitly into the definition of a distinguisher enemy. We could have included them by saying that the enemy at each status request, in addition to the status of z, also gets to execute the appropriate protocol playing the role of V_C or V_D. However, the success of such an enemy would imply the success of an enemy of our kind: we have demanded that the protocols be zero-knowledge, and so the executions could be replaced by simulations without affecting significantly the final output.

We also remark that we have not considered parallel executions of the verify and/or disavowal protocols. In theory, this can always be justified, if the signer simply refuses to execute more than one protocol at a time. Even if this is not done in practice, it does not seem to lead to problems for the concrete schemes we present: although zero-knowledge is generally not closed under parallel composition, the concrete protocols involved here can reasonably be conjectured to be secure, even when executed in parallel.

We can now finally state:

Definition 3. An undeniable signature scheme is said to be *signature indistinguishable* if no distinguisher enemy has success against it.

We now come to the other part of security. For this, we need a new kind of enemy:

Definition 4. A *signature enemy* E_S is a probabilistic polynomial time algorithm, which can be used in the following type of experiment:

1. GEN is executed on input 1^k, let the output be K_S, K_V, K_P. As input, E_S gets K_P and 1^k.
2. E_S may now make any number of *status requests* and *signature requests* (see Definition 1).
3. Let M be the set of messages occurring in status or signature requests done in step 2. Now E_S outputs a message $m_0 \notin M$, and a string z_0.

We must now define what it means that E_S has success:

Definition 5. Let $p_{\text{sig}}(k)$ be the probability that E_S outputs (m_0, z_0) such that $z_0 \in \text{VALID}(m_0, K_P)$. This probability is taken over the random choices made by E_D, GEN and SIGN.

E_S is *successful* against the scheme

$$(\text{GEN}, \text{SIGN}, (C, V_C), (D, V_D), \text{VALID}, \text{SIGN}_{\text{Sim}}),$$

if there is a polynomial P such that for infinitely many k,

$$p_{\text{sig}}(k) \geq 1/P(k).$$

Definition 6. An undeniable signature scheme is said to be *unforgeable* if no signature enemy has success against it.

Finally, we need to define the additional property that a convertible scheme should have. From an undeniable signature scheme **S** described by the 6-tuple $(\text{GEN}, \text{SIGN}, (C, V_C), (D, V_D), \text{VALID}, \text{SIGN}_{\text{Sim}})$, we can always build an ordinary signature scheme with secret key K_S and public key (K_V, K_P). Signatures are generated by running SIGN and can be verified by simulating the verification protocol. This requires no interaction, when K_V is known (although for practical schemes there may be more efficient ways to verify). We call this the *derived signature scheme* of **S**. In cases where $K_V = K_S$, the derived scheme is of course totally insecure. But for convertible schemes, we would like it to be secure in the standard sense of Goldwasser, Micali and Rivest (refer to [GMR88] for details):

Definition 7. An undeniable signature scheme is said to be *convertible*, if its derived signature scheme is not existentially forgeable under an adaptive chosen message attack.

To build a convertible scheme, it seems like a natural idea to generate an ordinary signature and encrypt it under some public-key probabilistic encryption scheme. Indeed, this idea can be quite easily proved to work, if the signature and encryption schemes are secure in a strong enough sense. However, even if those schemes were practical, the combined result will not be practical in general. This is because the only general way to build verification and disavowal protocols is by secure circuit evaluation, which will be polynomial time, but usually horrible in practice. Thus, to preserve efficiency in a practical sense a more careful way of doing the combination is needed. We show two examples of this in the following sections.

3 Two Schemes

This section presents two convertible undeniable signature schemes obtained from the El Gamal signature scheme [EG85]. In El Gamal signatures the public key is a triple (p, t, g, h), where p is a prime, t divides $p-1$ and g generates the subgroup, G, of Z_p^* of order t. Finally, $h = g^x \bmod p$, where $x \in \{0, 1, \ldots, t-1\}$ is the secret key. The signature on a message $m \in \{0, 1, \ldots, t-1\}$ is a pair $(r, s) \in G \times Z_t$ satisfying $g^m = h^r r^s \bmod p$ (when r is in the exponent, the binary representation of r is interpreted as a number in Z_t). A signature is made by choosing $b \in Z_t^*$ at random, computing $r = g^b \bmod p$ and finding s as the solution to the equation $m = rx + bs \bmod t$.

For security and efficiency reasons, a hash value of the message and not the message itself is usually signed. In the following it will therefore implicitly be assumed that m is the result of hashing the actual message using an agreed hash function. This also means that the message space of the two schemes presented below is $\{0,1\}^*$.

Both convertible undeniable signature schemes below are obtained by encrypting the second part of the signature (s). One scheme uses Rabin encryption [Rab79] and the other uses Diffie-Hellman encryption [DH76, EG85].

Some common notation will be used in addition to that already introduced. If a is an element of a well defined group, $\mathrm{ord}(a)$ denotes the order of a, $<a>$ denotes the subgroup generated by a, and $\log_a b$ denotes the discrete logarithm of $b \in <a>$ with respect to a.

Both schemes require that $p = \omega t + 1$, where t is of a special form and $\omega \in N$. One way to achieve this is to first generate t as required and then $p = \omega t + 1$ as small as possible. In [Wag79] it is argued that given a random t, p can be expected to be less than $t \log_2^2 t$.

3.1 Rabin Encryption of s

This scheme assumes that t is selected as a product of two large primes q_1 and q_2. Knowing the factorisation of t allows verification of signatures. Thus the scheme can be described as follows:

- The key generator, generates p and t, where t is the product of two k-bits primes q_1 and q_2. Next, using the factorisation of $p-1$, g of order t and $x \in Z_t^*$ are chosen, and $h = g^x \bmod p$ is computed. The public key is $K_P = (p, t, g, h)$, the secret key is $K_S = x$ and the verification key is $K_V = (q_1, q_2)$.
  The language, $\mathcal{L}$, of legal public keys depends on a parameter k_{min} and is defined as the set of tuples (p, t, g, h) such that $g^t = h^t = 1 \bmod p$ and all divisors of t have binary length at least k_{min}.
  Remark Membership of $\mathcal{L}$ can be verified in polynomial time, whenever k_{min} is logarithmic in k. $\mathcal{L}$ allows keys for which g and h have order less than t, and t is the product of several small primes.
- A signature on a message m is pair $(r, E(s))$ computed by making an El Gamal signature (r, s) on m and computing $E(s) = s^2 \bmod t$.
- Given a signature (r, E) on m, let u denote $g^m h^{-r} \bmod p$. Then the set of valid signatures on m is defined as follows:

$$\text{VALID}(m, K_P) = \left\{(r, E) \in G \times Z_t^* \mid \exists s \in Z_t^* : u = r^s \ \wedge\ r^E = u^s\right\}.$$

  Remark Clearly, $\text{SIGN}(m, x) \in \text{VALID}(m, K_P)$. On the other hand, if $(r, E) \in \text{VALID}(m, K_P)$, then $E = s^2 \bmod \text{ord}(r)$ for some s such that (r, s) is a signature on m. If the signature is constructed using SIGN then $\text{ord}(r)$ equals t, but in general they may be different. This will, however, not cause any security problems.
- The signature simulator selects $r \in G$ at random and $E(s)$ as a random quadratic residue modulo t.

To complete the description of the schemes, protocols for verifying and disavowing signatures must be given. The definition of $\text{VALID}(m, K_P)$ shows that these can be based on zero-knowledge proofs of equality and inequality of discrete logarithms.

For verification, the variant of Schnorr [Sch91] presented in [CP93] can be used to obtain a zero-knowledge proof for $\text{VALID}(m, K_P)$. If the challenge is selected from a suitable set (e.g. among 2^{20} possibilities) only two or three iterations are needed making the verification protocol practical, and the protocol can still be simulated efficiently. Another possibility is the cut-and-choose protocol of [CEG87]. However, the zero-knowledge protocol of [Cha91] does not work immediately, since the signer may cheat if u and r are in different subgroups.

The protocol for disavowal can be obtained by techniques similar to those used for denying signatures in [BCDP91].

3.2 Diffie-Hellman Encryption of s

This scheme assumes that t is a prime. The number s is then encrypted using Diffie-Hellman encryption modulo t. The scheme is defined as follows:

- The key generator, generates a k-bits prime t and a prime p of the form $p = \omega t + 1$. Next, g of order t and $x \in Z_t^*$ are chosen and $h = g^x$ mod

p is computed. Furthermore, a generator α of Z_t^* and $v \in \{0, 1, \ldots, t-1\}$ are selected and β is computed as $\alpha^v \bmod t$. The public key is $K_P = (p, t, g, h, \alpha, \beta)$, the secret key is $K_S = x$ and the verification key is $K_V = v$. The language, $\mathcal{L}$, of legal public keys, is defined as the following set of tuples $(p, t, g, h, \alpha, \beta)$ such that t is a prime, $g^t = h^t = 1 \bmod p$ and both α and β generate Z_t^*. The owner must publish the factorisation of $t-1$ allowing other parties to verify that α and β generate Z_t^*.
Remark Unlike the previous scheme, this one allows several persons to use the same p, t, g and α. If these numbers are published a priori as system parameters, the key generator just has to select x and v and compute h and β.

- A signature on a message m is a pair $(r, E(s, \rho))$ computed by making an El Gamal signature (r, s) on m, selecting $\rho \in Z_{t-1}$ at random and computing $E(s, \rho) = (\alpha^\rho, s\beta^\rho)$ modulo t.
- Given a signature (r, E) on m, let u denote $g^m h^{-r} \bmod p$ and let $E = (E_1, E_2)$. Then the set of valid signatures on m is defined as follows:

$$\text{VALID}(m, K_P) =$$

$$\left\{(r, E) \in G \times Z_t^* \times Z_t^* \mid \exists s \in Z_t : \log_\alpha \beta = \log_{E_1}(E_2 s^{-1}) \wedge u = r^s\right\}.$$

  Clearly, $\text{SIGN}(m, x) \in \text{VALID}(m, K_P)$.
- The signature simulator selects $r \in G$ at random and E as a pair of random elements of Z_t^*.

Next protocols for verifying and disavowing signatures are described. Given a possibly false signature (r, E) on m, let $E = (z_1, z_2)$ and let $u = g^m h^{-r} \bmod p$ as above. For verification the verifier must show that (z_1, z_2) encrypts a number, s such that $u = r^s \bmod p$ and for disavowal the denier must show that (z_1, z_2) encrypts a number s such that $u \neq r^s \bmod p$. An interactive bi-proof for this is depicted in Figure 1. A bi-proof system uses the same protocol for verification and disavowal. The verifier will accept the verification or the disavowal depending on the outcome of the protocol (see [FOO91]) for details).

This protocol requires a proof that E' is an encryption of ss' (see Figure 1 for the notation). Such a proof can for example be obtained using the efficient zero-knowledge proof in [Cha91]. However, in the case of verification the prover can do even better, by sending $\rho + \rho' \bmod t$. This is possible if the prover knows ρ (e.g., as a consequence of choosing ρ using a function from a family of pseudo random functions [GGM84]). In both cases the following holds:

Proposition 8. *If the protocol in Figure 1 is repeated $l = \Omega(k)$ times it is a perfectly zero-knowledge proof system for* $\text{VALID}(m, K_P)$. *In general a cheating prover can convince a verifier with probability at most 2^{-l}.*

Proof
Completeness and Soundness
Completeness is clear by inspection of the protocol.

The verifier first checks that $(r, E) \in G \times Z_t^* \times Z_t^*$. In particular this implies that r must have order t. If this fails the verifier will reject in case of verification and accept in case of disavowal.
Using v, the prover next computes s such that (z_1, z_2) is an encryption of s. Then the following is repeated l times:

1. The prover chooses $\rho' \in Z_{t-1}$ and $s' \in Z_t^*$ at random and sends $E' = \left(z_1\alpha^{\rho'}, z_2 s'\beta^{\rho'}\right)$ and $w = u^{s'} \bmod p$ to the verifier.
2. The verifier chooses $b \in \{0,1\}$ at random and sends b to the prover.
3. If $b = 0$ the prover sends back the pair (s', ρ'). Otherwise, if $b = 1$ the prover sends back $ss' \bmod t$ and proves (possibly interactively) that E' is an encryption of ss'.
4. If $b = 0$ the verifier checks that E' and w are computed as in Step 1. If $b = 1$ the verifier checks the proof of the decryption of E'. If this fails the verifier rejects the proof. Otherwise, if the prover is verifying a signature, the verifier accepts if

$$r^{ss'} = w \bmod p$$

and if the prover is disavowing a signature the verifier accepts if $r^{ss'} \neq w \bmod p$

Fig. 1. Bi-proof system for $\text{VALID}(m, K_P)$ (if $l = k$). The common input is (K_P, r, u, z_1, z_2) and the private input of the prover is v such that $\beta = \alpha^v \bmod t$.

For soundness first observe that we may assume that r has order t, since otherwise the verifier would immediately reject. We now show that if for at least one of the iterations, the prover is capable of satisfying the verifier for both $b = 0$ and $b = 1$, then the input signature is valid. This immediately implies that the prover can cheat with probability at most 2^{-l}. In the initial message of the round the prover sends an encryption (E_1', E_2') and a number w. A correct answer to $b = 0$ is a pair of numbers s', ρ' such that

$$(E_1', E_2') = (z_1\alpha^{\rho'}, z_2 s'\beta^{\rho'}) \text{ and } w = u^{s'}.$$

A correct answer to $b = 1$ is a number d for which $r^d = w$ for verification and $r^d \neq w$ for disavowal. Moreover we know that (E_1', E_2') encrypts d (since the prover decrypts it directly in the verification case), i.e. that for some γ

$$(E_1', E_2') = (\alpha^\gamma, d\beta^\gamma).$$

By putting the two equations on (E_1, E_2) together, we can obtain that $(z_1, z_2) = (\alpha^{\gamma-\rho'}, s'^{-1}d\beta^{\gamma-\rho'})$ where s'^{-1} denotes the multiplicative inverse modulo t. Since t is also the order of r, the fact that for verification $u^{s'} = w = r^d$ implies that $u = r^{s'^{-1}d}$. Thus we have shown that for verification, (z_1, z_2) encrypts the discrete log base r of u, which is the condition specified in the definition of $\text{VALID}(m, K_P)$.

Zero-Knowledge

We exhibit a simulator for one round of the verification case:

1. Choose c to be 0 or 1 at random.
2. If $c = 0$, follow the prover's algorithm for computing E' and $w = u^{s'}$ and send them to the verifier.
   If $c = 1$, choose $d \in Z_t^*$ at random and compute E' as an encryption of d. Compute $w = r^d$ and send E' and w to the verifier.
3. Receive b from the verifier.
4. If $c \neq b$, rewind the verifier and go to step 1.
   If $c = b = 0$, send s', ρ' to the verifier.
   If $c = b = 1$, send d to the verifier and convince him that E' encrypts d (following the prover's procedure for doing this).

To simulate l rounds, just repeat this simulation.

Observe that the prover's first message is always an encryption of a random number d, and a w such that $w = r^d$. Hence the simulators first message has the same distribution as the prover's and is in particular independent of c. Hence the expected number of rewinds is 2 and the complete simulation runs in expected linear time. It is easy to check that the answers generated by the simulator in step 4 have the same distribution as the prover's responses. Thus the simulation is perfect. □

When the protocol is used to deny signatures, it is only computationally zero-knowledge. The problem is that the for the case $b = 1$ it does not seem possible to come up with both the product ss' and $u^{s'}$ since $u^{1/s}$ is not known. Instead, this case is simulated by choosing ss' at random, making a random encryption of ss' and instead of $u^{s'}$ a random number in $<g>$ is selected. To show that this simulation works, the following assumption is necessary:

Assumption EDL If p, t, α and β are selected as described by GEN, then given a pair (a, b) of elements of Z_t^* it is not feasible to say if (a, b) is chosen at random or as a random pair satisfying $\log_\alpha \beta = \log_a b$.

This assumption has previously been used in [CvA90, BCDP91].

Proposition 9. *Under Assumption EDL, the protocol in Figure 1 is a computationally zero-knowledge proof system for the complement of* VALID(m, K_P).

Proof

Completeness and soundness

Again completeness is clear. For soundness first observe that we may assume that r has order t, since otherwise the verifier would immediately accept. As in the proof of Proposition 8 correct answers to both $b = 0$ and $b = 1$ allows us to write $(z_1, z_2) = (\alpha^{\gamma-\rho'}, s'^{-1} d \beta^{\gamma-\rho'})$ where s'^{-1} denotes the multiplicative inverse modulo t. We know that $u^{s'} = w \neq r^d$, whence $u \neq r^{s'^{-1} d}$. Since we still have that $s'^{-1} d$ is the number encrypted in (z_1, z_2), the condition for being in VALID(m, K_P) is not satisfied.

This shows that given that the prover cannot cheat in any of the proofs that (E_1', E_2') encrypts d, his chance of cheating is at most 2^{-l}. Let ϵ be the probability with which the prover can cheat in one such proof. Then the probability that he can cheat in any of the l proofs is at most $l\epsilon$. We may assume that ϵ is exponentially small in k - it may even be 0, if the prover can decrypt the pair directly. Therefore also $l\epsilon$ is exponentially small. This clearly implies that his overall chance is exponentially small in k (if we use $l = k$), and is in fact 2^{-l} if $\epsilon = 0$.

Zero-knowledge

The simulation works as follows:

1. Choose c to be 0 or 1 at random.
2. If $c = 0$, follow the prover's algorithm for computing E' and $w = u^{s'}$ and send them to the verifier.
   If $c = 1$, choose $d \in Z_t^*$ at random and compute E' as an encryption of d. Choose w at random and send E' and w to the verifier.
3. Receive b from the verifier.
4. If $c \neq b$, rewind the verifier and go to step 1.
   If $c = b = 0$, send s', ρ' to the verifier.
   If $c = b = 1$, send d to the verifier and convince him that E' encrypts d (following the prover's procedure for doing this).

To simulate l rounds, just repeat this simulation.

Let the simulators first message be the encryption (E_1', E_2') and the number w. Let s' be the number encrypted by the pair $(E_1' z_1^{-1}, E' z_2^{-1})$. Now, if $c = 0$, $w = u^{s'}$ while if $c = 1$, w is independent of s'. Distinguishing the two cases is at least as hard as deciding if $\log_\alpha(E_1' z_1^{-1}) = \log_\beta(s'^{-1} E_2' z_2^{-1})$. Hence if the probability that $b = c$ is significantly different from $1/2$, the verifier could be used to construct an algorithm contradicting assumption EDL. Hence, under EDL, the simulation runs in expected polynomial time.

This also shows that the cases $b = 1$ and $b = 0$ occur in the simulation with probabilities as in the real conversation except for a superpolynomially small error. Now note that the distribution of the simulation given that $b = 0$ is the same as for the conversation, whereas the distribution given that $b = 1$ is different from the conversation. However, by the same argument as before the two cannot be distinguished in polynomial time under assumption EDL. □

Remark The verification protocol is not as efficient as the one for the scheme based on Rabin encryption, however this scheme offers additional flexibility, such as

- The signer can convert single signatures (selective conversion as described in [BCDP91]) by releasing ρ.
- If the signer chooses a list of β-values, different β's can be used for different classes of signatures. All signatures in one class can then be converted by publishing the corresponding $\log_\alpha \beta$.

4 Security of the Schemes

This section analyses the security of the two schemes, with respect to forgeries and recognising signatures.

4.1 Forging Signatures

For both schemes it is sufficient (and necessary, if the signatures are converted) to show that the they are secure if the verification key is published.

Proposition 10. *The schemes using Rabin respectively Diffie-Hellman encryption are unforgeable, if the El Gamal scheme combined with the chosen hash function is secure against adaptively chosen message attacks for primes of the form $\omega t + 1$ where t is the product of two large known primes respectively t is a large known prime.*

Proof sketch
Given a method for forging signatures in one of the two schemes, signatures can be forged in the corresponding El Gamal scheme using the following adaptively chosen message attack:

1. Generate the verification key, and the corresponding part of the public key in the undeniable signature scheme.
2. Execute the attack against the undeniable scheme with the resulting public key. Whenever, an undeniable signature on a message, m, is requested in this attack, ask for an El Gamal signature on m and encrypt s to obtain the required undeniable signature. Whenever, the attacker asks if $z \in \text{VALID}(m, K_P)$ for some message m, this is answered by decrypting z and verifying the signature.
3. Finally, the attack outputs an undeniable signature on a new message, m_0. This signature can be made into an ordinary El Gamal signature by decrypting s.

□

Corollary 11. *Both schemes are convertible if the El Gamal schemes they are based on are secure against adaptively chosen message attacks.*

5 Signature Indistinguishability

Recall the definition of signature indistinguishability from Section 2. First, note that for both schemes, it is enough to argue indistinguishability in the case where the enemy knows K_S, as this can only make his task easier. In this case, being able to get signatures on chosen messages is of no additional help. In the following, an active, resp. passive attack is one where the enemy uses, resp. does not use the oracle for testing validity of signatures of his choice.

5.1 The Scheme using Rabin Encryption

When the enemy is given a purported signature (r, E) on m, all he knows is that if the signature is valid then the s determined by the equation $m - xr = bs \bmod t$ (where b is the discrete log of r base g) equals the s' determined by the equation

$s'^2 = E \bmod t$. If the enemy neither knows b nor the factorisation of t this seems to be difficult to decide.

In an active attack, the enemy may try to manipulate the equations in order to get the oracle to solve his problem. For example, if m' satisfies that $m - xr = m' - xr^{\rho^{-1}}$ for some ρ, then $(r^{\rho^{-1}}, E\rho^2) \in \text{VALID}(m', K_P)$ iff $(r, E) \in \text{VALID}(m, K_P)$. Recall, however, that m is actually a hash value, so that this attack is useless, unless the hash function can be inverted on m'.

We therefore conjecture that if discrete log mod p, factorisation of t, and inversion of the hash function are hard problems, then this scheme is signature indistinguishable.

5.2 The Scheme using Diffie-Hellman Encryption

In this scheme, the enemy must decide, given $m, r, (E_1, E_2)$ whether the s determined by the equation $m - xr = bs \bmod t$ (where b is the discrete log of r base g), is also the value encrypted by the pair (E_1, E_2). Even if the enemy knows b, and hence s, he is left with the problem of deciding whether $\log_\alpha \beta = \log_{E_1}(E_2 s^{-1})$. Hence, in a passive attack, the enemy's problem is at least as hard as deciding equality of discrete logs (assumption EDL).

In an active attack, the enemy may try as above to exploit the multiplicative properties of the encryption. Indeed, manipulations similar to the example above are possible. Once again, however, this seems useless, unless the hash function can be inverted. We therefore conjecture that if assumption EDL holds, and the hash function used is one-way, this scheme is signature indistinguishable. We remark that exactly the same assumptions were used for the scheme from [CvA90], the oldest surviving undeniable signature scheme.

References

[BCDP91] J. Boyar, D. Chaum, I. Damgård, and T. Pedersen. Convertible Undeniable Signatures. In *Advances in Cryptology - proceedings of CRYPTO 90*, Lecture Notes in Computer Science, pages 189 – 205. Springer-Verlag, 1991.

[CEG87] D. Chaum, J.-H. Evertse, and J. van de Graaf. An Improved Protocol for Demonstrating Possession of a Discrete Logarithm and some Generalizations. In *Advances in Cryptology - proceedings of EUROCRYPT 87*, Lecture Notes in Computer Science, pages 127–141, 1987.

[Cha91] D. Chaum. Zero-Knowledge Undeniable Signatures. In *Advances in Cryptology - proceedings of EUROCRYPT 90*, Lecture Notes in Computer Science, pages 458 – 464. Springer Verlag, 1991.

[CP93] D. Chaum and T.P. Pedersen. Wallet Databases with Observers. In *Advances in Cryptology - proceedings of CRYPTO 92*, Lecture Notes in Computer Science, pages 89–105. Springer-Verlag, 1993.

[CvA90] D. Chaum and H. van Antwerpen. Undeniable Signatures. In *Advances in Cryptology - proceedings of CRYPTO 89*, Lecture Notes in Computer Science, pages 212–216. Springer Verlag, 1990.

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Trans. Inform. Theory*, IT–22(6):644–654, November 1976.

[DY91] Y. Desmedt and M. Yung. Weaknesses of Undeniable Signature Schemes. In *Advances in Cryptology - proceedings of EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 205–220. Springer-Verlag, 1991.

[EG85] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology - proceedings of CRYPTO 84*, Lecture Notes in Computer Science, pages 10 –18. Springer-Verlag, 1985.

[FOO91] A. Fujioka, T. Okamoto and K. Ohta. Interactive Bi-Proof Systems and Undeniable Signature Schemes. In *Advances in Cryptology - proceedings of EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 243–256. Springer-Verlag, 1991.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. In *Proceedings of the 25th IEEE Symposium on the Foundations of Computer Science*, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen Message Attack. *SIAM Journal on Computing*, 17(2):281 - 308, April 1988.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems. *SIAM Journal of Computation*, 18(1):186–208, 1989.

[JKR96] M. Jakobsson, K. Sako and R. Impagliazzo: Designated Verifier Proofs and Their Applications, 1996. These proceedings.

[Mic90] S. Micali, August 1990. Personal communication.

[Mic95] M. Michels. Breaking and Repairing a Convertible Undeniable Signature Scheme. Technical Report TR-95-10-D, University of Technology, Chemnitz-Zwickau, June 1995. To appear at ACM Security, March 1996.

[Rab79] M. O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, MIT, January 1979.

[Sch91] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161-174, 1991.

[Wag79] S. S. Wagstaff Jr. Greatest of the Least Primes in Arithmetic Progression Having a Given Modulus. *Mathematics of Computation*, 33(147):1073 – 1080, July 1979.