Convertible Undeniable Signatures

Joan Boyar*
Aarhus University

David Chaum
CWI

Ivan Damgård
Aarhus University

Torben Pedersen
Aarhus University

Abstract

We introduce a new concept called *convertible undeniable signature schemes.* In these schemes, release of a single bit string by the signer turns all of his signatures, which were originally undeniable signatures, into ordinary digital signatures. We prove that the existence of such schemes is implied by the existence of digital signature schemes. Then, looking at the problem more practically, we present a very efficient convertible undeniable signature scheme. This scheme has the added benefit that signatures can also be selectively converted.

1 Introduction

Undeniable signatures were introduced in [CvA90]. For certain applications, these signatures are preferable to ordinary digital signatures because simply copying a signature does not produce something which can be directly verified. Instead, signatures are verified via a protocol between the signer and verifier, so the cooperation of the signer is necessary. The signer is not, however, allowed to deny a signature simply by refusing to participate in the protocol; there must also be a protocol which the signer can use in order to deny an invalid signature.

If, for example, a software company wanted to certify that they had provided a certain program, they could sign it using an undeniable signature. Only someone who had directly purchased the software from that company could verify the signature and be certain that no viruses had been introduced. However, if the software company sold programs which contained bugs, they should be unable to deny their signatures afterwards.

In addition to the properties of undeniable signatures described above, it could be useful if there were some secret information, which the signer could release at some point after signing, which would turn the undeniable signatures into ordinary digital signatures. Thus these signatures could be verified without the aid of the signer,

*Supported in part by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (Project ALCOM).

A.J. Menezes and S.A. Vanstone (Eds.): Advances in Cryptology - CRYPTO '90, LNCS 537, pp. 189-205, 1991.

but they should still be difficult to forge. We will call such signatures *convertible undeniable signatures.*

A first advantage in the software company example of using convertible undeniable signatures is that several employees of the company could be given the ability to verify signatures, without being able to sign messages.[1] This would pose a smaller security risk than if the entire secret key to the system was distributed this way.

Secondly, if the company later goes bankrupt (or dissolves for any reason), it can release the secret information needed to verify signatures, so the software can still be used safely. At this point, the company would no longer be protected against the pirating of its software, but it would no longer care much about it either.

Undeniable signatures could also be useful in any situation in which an individual wishes to sign a document, but does not want the press to be able to prove that he signed that document. Suppose further that this individual must allow some other party to prove later that the signature is valid, without allowing this other party to forge signatures. An example of such a situation is a last will and testament which contains instructions that the signer would not want made public. After the signer's death, it is important that the signer's attorney be capable of validating the signature. The attorney should not, however, be capable of creating a new will and forging the deceased's signature. If a convertible undeniable signature is used, the attorney can be given the secret information which converts the undeniable signature into a digital signature. Then the attorney could act on the signer's behalf, verifying and denying signatures.

To summarize, for each signer there should be a public key K_P and two private keys K_{S1} and K_{S2}. The first private key K_{S1} should never be released; the signer uses it to produce signatures. The second private key K_{S2} may be released to convert the undeniable signatures into ordinary digital signatures.

- There should be an efficient algorithm for producing signatures, using the private keys K_{S1} and K_{S2}.
- There should be efficient protocols (efficient for both the signer and the verifier), one for verifying valid signatures and one for denying invalid signatures. If the signer participates in one of these protocols successfully, the verifier should be convinced that the signer had a negligible probability of cheating.
- After the undeniable signatures have been converted to digital signatures, it should be easy to verify them, so there should be an efficient algorithm which, given K_P and K_{S2}, can be used to verify valid signatures.

In order for a convertible undeniable signature scheme to be secure, it should also have the following three properties:

- The verifier should be unable to forge the signer's signature on any message for which it has not already seen a signature, even though
  1 The verifier may be able to get the signer to sign arbitrary messages.

[1]The key which converts the undeniable signatures into digital signatures can be used in verifying and denying protocols which are zero-knowledge proofs based on the circuits for verifying and denying the underlying digital signatures.

2 The verifier may be able to get the signer to enter into the verifying or denying protocol (whichever is appropriate) for arbitrary "signed" messages.

3 The second private key K_{S2} may be available. One may assume, however, that after K_{S2} is available, the prover refuses to participate in any more verifying or denying protocols.

- The verifier should be unable to produce a string for which the signer, with the correct public key K_P, is unable to either verify or deny that the string is a signature, even though
  1 The verifier may be able to get the signer to sign arbitrary messages.
  2 The verifier may be able to get the signer to enter into the verifying or denying protocol (whichever is appropriate) for arbitrary "signed" messages.
- The verifier should be unable to distinguish between valid and invalid signatures, with any significant advantage, without entering into one of these protocols with the signer. See section 3.1 for a formalization of this.

In some applications of convertible undeniable signatures, including that of the last will and testament, mentioned above, one might prefer to convert only selected undeniable signatures into digital signatures. (The signer may not wish everything he ever signed to be public, even after his death.) When a scheme allows this, we will say that it is a *selectively convertible undeniable signature scheme.*

The next section in this paper discusses previous work on undeniable signatures. In section 3, we first present a formal definition for undeniability, and then we present a selectively convertible undeniable signature scheme, based on the sole assumption that a secure digital signature scheme exists. The reader who is more interested in practical than theoretical results could skip that section and continue with section 4, in which we present an efficient selectively convertible undeniable signature scheme based on El Gamal signatures. The final section of the paper is a summary.

2 Related Work

In [CvA90], undeniable signatures were presented for the first time, and an implementation was described, based on the difficulty of computing discrete logarithms in a cyclic group of prime order. In this scheme, the public key has the form (g, g^x), where g is a generator of the group, and x is the secret key. For a message m, the signature has the form m^x.

In [Cha90] zero-knowledge denial and verification protocols were presented for this scheme. Its security and efficiency is based on the fact that the group order is prime and public, and on the assumption that all messages are bit strings representing elements in the group (or there is an easy way of checking that a message is in the group).

These conditions can be met, for example, by using the subgroup of index 2 of $\mathbb{Z}_p^*$, where p is a prime and $p - 1 = 2q$ for q prime. It is natural to try to generalize this scheme to other groups, for example by using a composite modulus, or by using the group on an elliptic curve. Such generalizations could potentially be more secure and

could (in the composite modulus case, where one could release x^{-1} without revealing x) produce a convertible scheme.

Such a straightforward generalization, however, immediately runs into problems, most notably the fact that there may be no way (even for the signer) to tell whether a given message represents an element in the cyclic subgroup we are using.

The scheme we present in section 4 avoids these problems, is almost as efficient as the scheme from [CvA90] when used in the same group,[2] and could be generalized to any group in which discrete log is a hard problem.

3 Theoretical Results

3.1 A definition of undeniability

Undeniable signatures differ from ordinary digital signatures in that, given an undeniable signature, the verifier should be unable to distinguish between valid and invalid signatures, with any significant advantage, without entering into either a verifying or denying protocol with the signer. In order to make this more precise, we need to have a simulator which can produce fake signatures which cannot be distinguished from valid signatures. This signature simulator should also be able to produce transcripts of verifying protocols which are indistinguishable from true transcripts so that the existence of a good transcript does not prove that a signature is valid. This is important in some applications such as that of the software company wishing to protect against the piracy of its software. In addition, we must ensure that this task of distinguishing between fake and valid signatures does not become significantly easier if the verifier enters into verifying and denying protocols for other messages and signatures.

Definition 3.1
A *signature simulator, relative to a verifier* V, is a probabilistic polynomial time algorithm, which, when given a message m, outputs a string $Fake(m)$ and a simulated transcript $FakeT(Fake(m), m)$ of a verifying conversation between the signer and V, "proving" that $Fake(m)$ is a valid signature.

Definition 3.2
The *signature oracle* O, *relative to a verifier* V, receives a message m as input and flips a fair coin to decide whether to

1. run the signature simulator relative to V and give the simulator's output, or
2. output a random valid signature, $Sign(m)$, and a transcript $ValidT(Sign(m), m)$, chosen randomly from the distribution of transcripts from verifying conversations, involving the true signer and the verifier V, proving that $Sign(m)$ is a valid signature.

[2]The public keys in the scheme presented here are a little longer than those in [CvA90], and slightly more computation is required

Definition 3.3
The probabilistic polynomial time *distinguisher* D, attempting to distinguish between the fake and valid outputs of the oracle is allowed to:

1 Choose a message m and give that to the signature oracle O.

2 Observe an output (s, T) from O. This pair (s, T) is either from the set FF, which contains forgery-transcript pairs of the form

$$(Fake(m), FakeT(Fake(m), m))$$

or from the set SV, which signature-transcript pairs of the form

$$(Sign(m), ValidT(Sign(m), m)).$$

3 Interact with the true signer, obtaining some valid signatures on messages in a set M', with $m \notin M'$.

4 Interact with the true signer in verifying protocols for messages in the set M' and signatures in the set S', and in denying protocols for messages in the set M'' and strings in a set S'' created by D, with $m \notin M'$, $m \notin M''$, $s \notin S'$, and $s \notin S''$.

Definition 3.4
Let $D(s, T, m)$ denote D's output when its input is the possible signature s, for message m, and possible transcript T. Let n be the security parameter. Then, a signature system is *undeniable* if, for any verifier V, there exists a signature simulator relative to V such that, for any polynomial time distinguisher D, and for any constant c, the following holds for n sufficiently large:

$$|Prob[D(s, T, m) = 1 : (s, T) \in SV] - Prob[D(s, T, m) = 1 : (s, T) \in FF]| < 1/n^c.$$

In some applications it would be useful for the verifying and denying protocols to be more symmetric in that transcripts of denying protocols should not be transferable, just as transcripts for verifying protocols should not be transferable. If this is the case, we have *symmetric undeniable signatures.* To make this more formal, one defines a *denial simulator relative to a verifier* V similarly to the signature simulator in definition 3.1, except that the simulator is either given the message m and a valid signature on m, or it is just given the message m. If the simulator is given m and a valid signature, it should output a fake transcript of a denying conversation between the signer and V "proving" that the signature is a forgery. If the simulator is only given m as input, it should produce a forged signature $Fake(m)$. The *denial oracle* should flip a fair coin to decide whether to

1 give the denial simulator m and a valid signature $Sign(m)$, and output $Sign(m)$ and the fake transcript produced by the denial simulator, or

2 give the denial simulator just the message m, and output the forgery $Fake(m)$ produced by the denial simulator, along with a random valid transcript of an execution of the denial protocol for $Fake(m)$ with the true prover and V.

The distinguisher would be similar to that in definition 3.3, except that the oracle used would be the denial oracle.

All of the signature schemes we present in this paper are, in fact, symmetric undeniable signature schemes, since all of our protocols are zero-knowledge. Since these protocols are zero-knowledge, to show that the schemes are undeniable, it is only necessary to prove that there is a signature simulator S_1 which can produce false signatures $Fake(m)$ that are indistinguishable from real signatures $Sign(m)$. To see this, let S_2 be the simulator guaranteed by the fact that the verifying protocol is zero-knowledge. Thus S_2 produces valid looking transcripts when given $(m, Sign(m))$ as input. But it must also do so when given $(m, Fake(m))$ as input since otherwise we would have a contradiction with the property of S_1. Thus running S_1 and then running S_2 on the output of S_1 will produce the simulator required by Definition 3.1. It is also clear from the zero-knowledge property that entering into verifying or denying protocols concerning other messages or signatures will not help the distinguisher.

Note that if the verifying and denying protocols are zero-knowledge, then protocol transcripts will not help an enemy either to forge a signature on any message for which it has not already seen a signature or to produce a string for which the signer is unable to either verify or deny that the string is signature.

It appears, at first glance, as if the above definition of undeniability requires that the verifying protocol be zero-knowledge. In fact, this does not seem to be the case. The difference is that we allow the signature simulator to create its own forgery, so it is conceivable that, although it can produce transcripts when it creates the forgery, it might be unable to produce good transcripts when it is just given a valid signature.

3.2 Existence of convertible undeniable signature schemes

In this section, we will discuss which assumptions are sufficient for the construction of schemes of the type in which we are interested. If one makes the assumption that a secure digital signature scheme exists, then we will see that it is quite easy to design a convertible undeniable signature scheme. Thus, by the result of [Rom90], it is sufficient to assume the existence of a one-way function.

By a "secure digital signature scheme" we mean one which is "not existentially forgeable under an adaptive chosen plaintext attack", i.e. even if an enemy can get signatures on messages of his choice, he cannot sign any message that has not been signed by the signer (see [GMR88]).

Theorem 3.5
There exists a secure selectively convertible undeniable signature scheme if and only if there exists a secure digital signature scheme.

Proof sketch
First, we remark that if a convertible undeniable signature scheme exists, then we trivially have an ordinary signature scheme by releasing K_{S2} immediately.

Conversely, to set up a digital signature scheme, there must be a polynomial time probabilistic algorithm which on input a random string r, produces a pair (P, S),

where P is a public key, and S is the matching secret key. It is intuitively obvious that the mapping from r to P must be a one-way function if the scheme is secure. A formal proof can be derived by using the scheme to build a secure identification protocol and using the result of [IL89].

Thus the existence of a secure digital signature scheme implies the existence of a one-way function, which by [GL89] and [ILL89] in turn implies the existence of pseudorandom generators, and hence by [GGM84] the existence of a pseudorandom function family. This is a parameterized family of functions $\{f_K\}$, where the parameter K can be thought of as a key. To a polynomially bounded enemy who does not know the key, images $f_K(x)$ appear to be totally random values with no correlation to x, even if the enemy gets to choose x.

In addition to pseudorandom functions, we will also need bit commitments. A bit commitment scheme may be thought of as a function BC that takes as input the bit string to be committed to, B, and some random input R. From this, one can compute the commitment $BC(B,R)$. Given only the commitment, it is "hard" to guess B better than at random, but it is also "hard" for the committer to change his mind, i.e. find $R, R', B \neq B'$, such that $BC(B,R) = BC(B',R')$. Given R, however, the commitment can be opened, i.e. B can be computed.

This description is actually a simplification - some commitment schemes require interaction - and commitments of the form just described require the existence of 1-1 one-way functions. We will assume first that bit commitments have this simple form, and describe later how to get rid of the 1-1 assumption.

Let P, S be a user's public and secret key for the signature scheme we are given and let $S(M)$ denote the signature of M using the secret key, S. We establish the undeniable system as follows: the user's public key is $K_P = (P, BC(K,R))$, where K is a key for the pseudorandom function family. The first private key is $K_{S1} = S$, and the second is $K_{S2} = R$. A signature on message M has the following form: $sign(M) = BC(S(M), f_K(M))$.

The protocols for verifying and denying signatures can be constructed from a circuit that works as follows (see figure 1): Using R, it will open the commitment to K, from which it computes $f_K(M)$. With this value, it opens the commitment to $S(M)$, and finally it checks this signature on M using the public key P. The circuit gives three bits $b1, b2, b3$ as output. They are defined to be 1, if the opening of the two commitments and the signature check, respectively, was successful.

By the general protocols of [BCC88] [IY88], the signer can now convince anyone of the value of any boolean function of $b1, b2, b3$ in zero-knowledge, in particular without revealing R. If he wants to verify a signature, he convinces the verifier that $b1 \wedge b2 \wedge b3 = 1$, if he wants to deny a signature, he convinces the verifier that $b1 \wedge ((\neg b2) \vee (\neg b3)) = 1$.

The scheme is secure against forgery because, since the signer chooses R independently of S, any forgery of the undeniable signatures could be used to forge signatures in the original signature scheme.

The scheme is undeniable because the signature simulator need only make a bit commitment to a random string of the correct length. If the bit commitment scheme and the pseudorandom function are secure, it should be impossible to distinguish

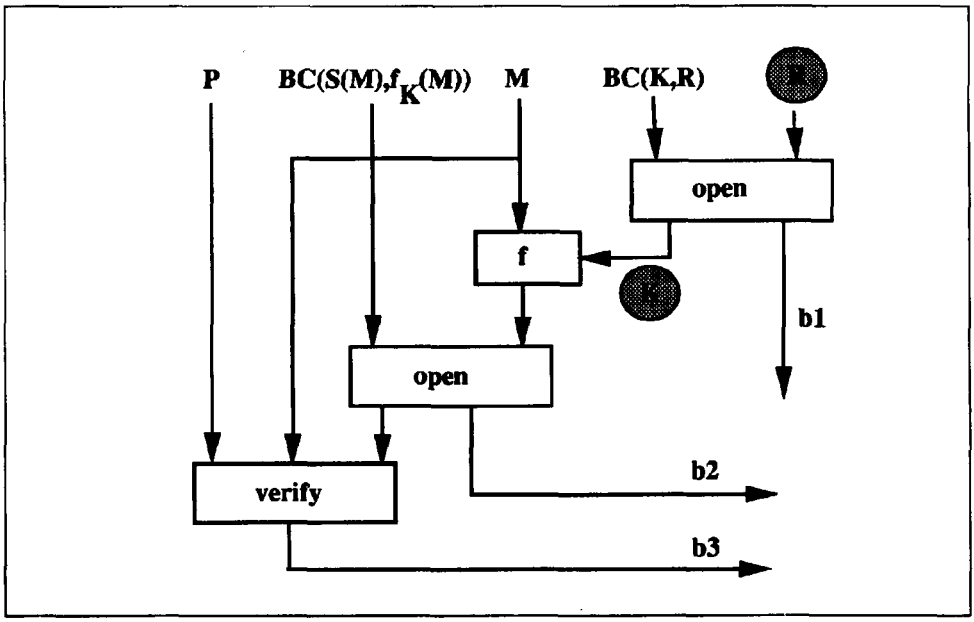

Figure 1: Circuit for verifying or denying signatures.

between this and a valid signature. The existence of this signature simulator, together with the fact that the protocols are zero-knowledge, is sufficient to prove that the scheme is undeniable.

The scheme is convertible, since the release of R enables the computation of K, and therefore all commitments to signatures can be opened. It is also selectively convertible, since for a signed message M, one can release $f_K(M)$. This allows computation of $S(M)$ from $sign(M)$, but by the properties of pseudorandom functions it does not help in computing any other function values.

Finally we address the problem of basing the scheme on *any* one-way function. By the result of Naor, [Nao90], one can build bit commitments from any one-way function, but this requires interaction: the verifier sends a random string R_V, and the signer/prover responds with the commitment. The randomness of R_V is necessary to ensure that the prover is actually committed, but it does not affect the secrecy of the bits committed to. We will use this scheme for the commitment to K; for this commitment, R_V can be supplied by a trusted key-center, which will usually have to exist to guarantee the authenticity of the public keys. But any mutually trusted source of random bits (such as a multiparty coin-flipping protocol) would suffice here. Once K is committed to, we do not need Naor's scheme any more. For the commitments to signatures, we can use the "hard-core" bits of the one-way function (see [GL89]), because the random input is now determined by K. ∎

The conditions needed for the construction of an undeniable signature scheme are no weaker than the necessary conditions for the construction of a convertible

undeniable signature scheme as shown in the following corollary.

Corollary 3.6
There exists a secure undeniable signature system if and only if there exists a one-way function.

Proof sketch
If undeniable signatures exist, one can prove the existence of a one-way function in the same way as in the proof of theorem of theorem 3.5.

The converse follows from the previous theorem and the result of [Rom90] that secure digital signatures exist if one-way functions exist. ■
Independently, this corollary has earlier been proven by Micali ([Mic90]).

4 A Practical Solution

In this section we show how the El Gamal signature scheme [EG85] can be changed into a convertible undeniable signature scheme.

First some notation. Let G be a group and let g be an element in this group. By $<g>$ we denote the subgroup of G generated by g, and by $|H|$ the order of the subgroup H ($|g| = |<g>|$). We use $DL(h,g)$ as shorthand for the discrete logarithm of h with respect to g. Thus $g^{DL(h,g)} = h$.

Let p be a prime such that it is hard to compute discrete logarithms in $\mathbb{Z}_p^*$ and let q be a large prime divisor of $p-1$ such that everybody knows q. In the following all computations are in $\mathbb{Z}_p^*$ unless otherwise specified.

Let α be a generator of the subgroup of $\mathbb{Z}_p^*$ of order q. The private key in the El Gamal signature scheme is a number x between 1 and q and the public key is $y = \alpha^x$. The signature on a message m ($1 \leq m < q$) is a pair (r,s) satisfying

$$\alpha^m = y^r r^s.$$

This signature is constructed by choosing $k \in [1, q-1]$ and computing r and s by

$$r = \alpha^k$$

and

$$s = k^{-1}(m - xr) \bmod q.$$

4.1 The Scheme

Using the El Gamal signature scheme, we can construct an undeniable signature scheme as follows. As before let α generate a subgroup of $\mathbb{Z}_p^*$ of prime order q. The private keys are

$$K_{S1} = x \text{ and } K_{S2} = z, \quad 1 < x, z < q$$

and the public key is

$$K_P = (\alpha, y, u), \text{ where } y = \alpha^x \text{ and } u = \alpha^z.$$

When the signer makes public $K_P = (\alpha, y, u)$, the receiver, which could be a key center or a user, should verify that $\alpha^q = y^q = u^q = 1$ and that none of the components of K_P is the identity. By the properties of $\mathbb{Z}_p^*$, this proves that α, y and u generate the same group.

The signature on the message m is $sign(m) = (\alpha^t, r, s)$, where (r, s) is the El Gamal signature on $\alpha^t tzm \bmod q$ (in this product α^t is considered to be a representation of an element in $\mathbb{Z}_q$).

In the El Gamal scheme, it is possible for a forger to construct a signature (r', s') on a message m'. It is very unlikely that m' is meaningful, but this attack implies that the El Gamal scheme must be used together with a hash function. The new scheme also requires a hash function, but we will not mention it when describing the scheme.

The triple (T, r, s) is a legal signature on a message m, if and only if

$$(T^{Tm})^z = y^r r^s$$

(whenever T is in the exponent, it is considered to be an element of $\mathbb{Z}_q$). Throughout this section we will use v to denote $y^r r^s$ and w to denote T^{Tm}. Thus verifying a signature (T, r, s) on m is equivalent to deciding if $DL(v, w)$ equals $DL(u, \alpha)$.

The signer can prove that (T, r, s) is a legal signature on the message m as follows: (S is the signer and V, the verifier)

PROTOCOL VERIFY SIGNATURE

1. S and V compute $w = T^{Tm}$ and $v = y^r r^s$.
2. S proves that $DL(v, w) = DL(u, \alpha)$ using the protocol for simultaneous discrete logarithm shown in figure 2 (see also [Cha90]).
3. V accepts the signature if and only if it accepts the proof.

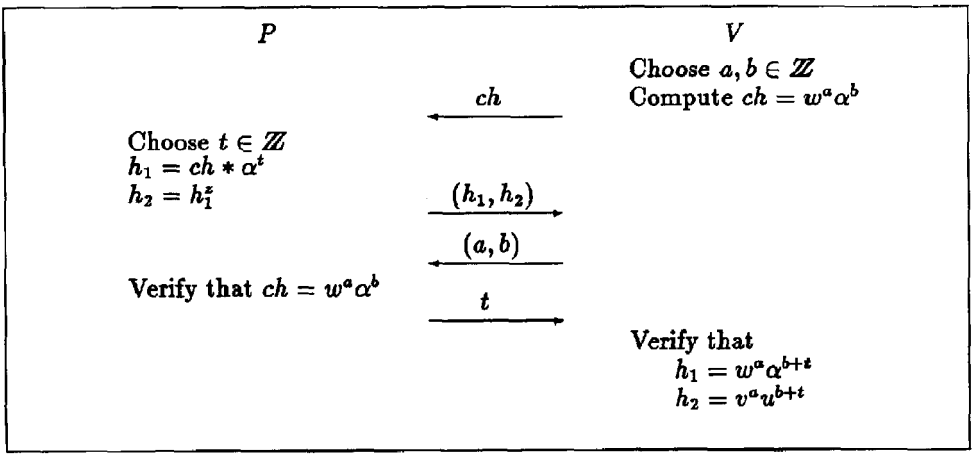

Figure 2: Proof that $DL(v, w)$ equals $DL(u, \alpha)$. The prover knows $z = DL(u, \alpha)$.

Lemma 4.1
If (T, r, s) is a legal signature on m, the following hold

1. The verifier always accepts.
2. PROTOCOL VERIFY SIGNATURE is zero-knowledge.

If (T, r, s) is a false signature, the verifier accepts with negligible probability (in the length of q).

Proof
It is sufficient to prove that the protocol in figure 2 is a zero-knowledge proof system with perfect completeness and negligible probability of the verifier accepting a false claim.

If $v = w^z$, the verifier will always accept.

Now assume that $w^z \neq v$.
If $w \neq 1$, w generates $<\alpha>$. For each value of $a \in \{0, \ldots, q-1\}$, there is exactly one value of b giving the same challenge. Thus ch contains no information about a. Now assume that the prover can find a pair (h_1, h_2) and t_1 and t_2 so that

$$h_1 = w^{a_1}\alpha^{b_1+t_1} = w^{a_2}\alpha^{b_2+t_2}$$

and

$$h_2 = v^{a_1}u^{b_1+t_1} = v^{a_2}u^{b_2+t_2}$$

where $a_1 \neq a_2$. Then we would have

$$\begin{aligned} DL(v,u) &= (a_1 - a_2)^{-1}((b_2 - b_1) + (t_2 - t_1)) \bmod q \\ &= DL(w,\alpha) \end{aligned}$$

and therefore $DL(v,w) = DL(u,\alpha)$, which is a contradiction. Hence S has probability at most $1/q$ of finding a pair (h_1, h_2) that V will accept.

The protocol is zero-knowledge, because for any verifier V', it can be simulated as follows

1. Get a challenge, ch, from V'.
2. Choose e and compute $h_1' = \alpha^e$ and $h_2' = u^e$.
3. Get (a, b) from the verifier.
   If $ch \neq w^a\alpha^b$, stop; and otherwise goto 4.
4. Rewind V' to after the challenge is sent.
   Choose t and compute $h_1 = w^a\alpha^{b+t}$ and $h_2 = v^a u^{b+t}$.
5. Get (a', b') from the verifier.
   If $ch = w^{a'}\alpha^{b'}$, send t to the verifier; and otherwise goto 4.

This simulation works because

- The verifier cannot find two different pairs (a_1, b_1) and (a_2, b_2) resulting in the same challenge without finding the discrete logarithm of w with respect to α.
- The first pair (h_1', h_2') has the same distribution as a pair (h_1, h_2) from the honest prover.

■

If (T, r, s) is a false signature, the simulation still works and it is impossible to tell that it is a simulation with a false signature unless one can distinguish legal signatures from invalid signatures. This property ensures that a transcript from an execution of PROTOCOL VERIFY SIGNATURE cannot be used as a proof of the validity of a signature.

Before presenting the protocol for denying false signatures, we consider the general problem of proving that $DL(v, w) \neq DL(u, \alpha)$. Knowing $z = DL(u, \alpha)$, a polynomial time prover can demonstrate that these logarithms are different as shown in figure 3. In this protocol, $BC(\gamma, R)$ denotes a bit commitment to the bit γ using random input R. The blob is opened by revealing R.

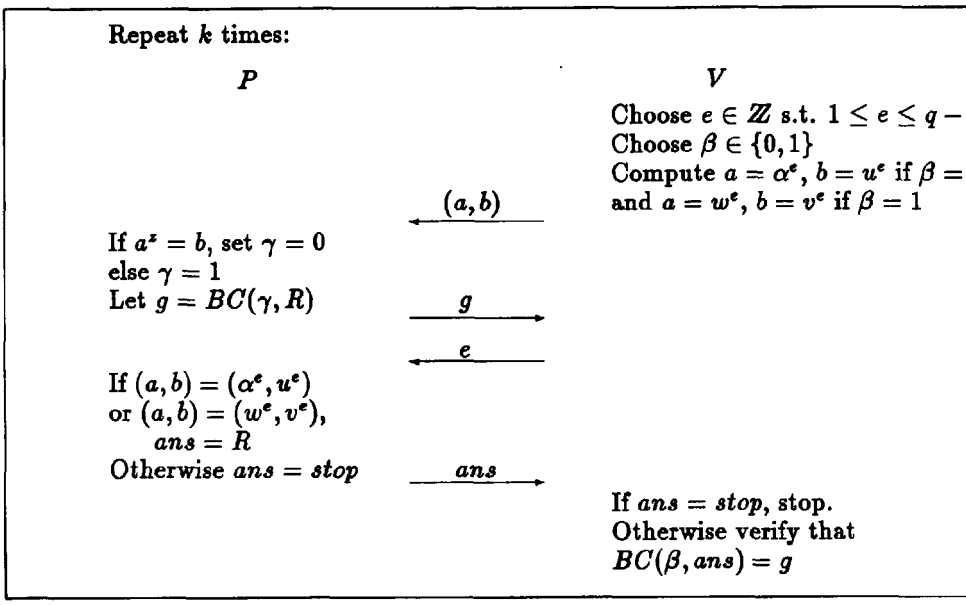

Figure 3: Proof that $DL(u, \alpha) \neq DL(v, w)$. The prover knows $z = DL(u, \alpha)$.

Lemma 4.2
If $DL(v, w) \neq DL(u, \alpha) \bmod q$, the verifier will accept with probability 1.
If $DL(v, w) = DL(u, \alpha) \bmod q$, the verifier will accept with probability 2^{-k}.
If the blobs are perfect, the protocol is perfect zero-knowledge even when the k rounds are run in parallel.

Proof
In the case where $DL(v,w) \neq DL(u,\alpha)$, the prover will always find $\gamma = \beta$. If $\beta = 0$,

$$a^z = \alpha^{ez} = u^e = b$$

and thus $\gamma = 0$. If $\beta = 1$ and $a^z = b$, we have

$$w^{ez} = a^z = b = v^e.$$

Since e is relatively prime to q, this implies that $w^z = v$ and thus $DL(v,w) = z = DL(u,\alpha)$. A contradiction.

If $DL(u,\alpha) = DL(v,w)$, the pair (a,b) contains no Shannon information about β, because w generates $<\alpha>$. Therefore, the prover can do no better than trying to guess β.

If the blobs are perfect, the protocol can be simulated using the same technique as in the proof of lemma 4.1. Therefore it is perfect zero-knowledge in this case. In addition, one can use the techniques of [GMW86] (graph non-isomorphism) to show that the protocol is also perfect zero-knowledge when the k rounds are run in parallel. ■

If the blobs are not perfect in the sense that encryptions of 0 are only computationally indistinguishable from blobs containing 1, a proof such as that in [BDLP89] shows that the protocol is computational zero-knowledge (also when run in parallel).

The above protocol is quite straightforward. An alternate protocol is given in [Cha90]. Chaum's protocol is more efficient in terms of the number of bits exchanged (or the number of rounds if the protocol is executed sequentially), but there is a trade-off requiring more computation on the part of the signer. In order to save a factor of $O(t)$ bits, the signer creates a table of 2^t powers of a specific element, and sorts the table. Extra computation would usually be much less expensive than communicating extra bits, so Chaum's protocol can be a big improvement over the one presented here.

With the proof system from figure 3, it is pretty straightforward to construct

PROTOCOL DENY SIGNATURE

1. S and V both compute $w = T^{Tm}$ and $v = y^r r^s$.
2. S proves that $DL(u,\alpha) \neq DL(v,w)$ using the protocol in figure 3.
3. S accepts the denial if and only if it accepts the proof.

Lemma 4.3
If (T,r,s) is a false signature on m, the verifier will always accept the denial.
If (T,r,s) is a legal signature on m, the verifier will reject the denial with high probability no matter what the signer does.
PROTOCOL DENY SIGNATURE is perfect or computational zero-knowledge depending on whether or not perfect blobs are used.

Proof
Follows from lemma 4.2. ■

4.2 Conversion of all signatures

As mentioned in the introduction, an undeniable signature is converted to an ordinary signature by releasing K_{S2}. Knowing $K_{S2} = z$ and K_P, everybody can verify a signature (T, r, s) on the message m by computing $(T^{Tm})^z$ and verifying that it equals $y^r r^s$. Thus when z is released, all previous and future signatures are equivalent to El Gamal signatures. Since anyone knowing z could have played the role of the prover in the previous executions of the protocols for verifying and denying signatures, the transcripts of these executions cannot help a forger (see also the subsection about security) after z is released.

4.3 Selective conversion

Knowing t such that $T = \alpha^t$, anyone can check that (T, r, s) is a signature on m by verifying that

$$T = \alpha^t \text{ and } u^{tTm} = y^r r^s.$$

Therefore, a single signature can be converted to an ordinary digital signature by releasing t. This method of converting signatures requires that the signer remembers the t used to construct the signature on m. This is most conveniently done by choosing a key k to a pseudorandom function f_k (see [GGM84]) and then computing t as $f_k(m)$. The properties of families of pseudorandom functions guarantee that, given polynomially many pairs $(m_i, f_k(m_i))$, it is infeasible to find $f_k(m)$ for a message $m \neq m_i$. Therefore, conversion of any polynomial number of signatures cannot affect the undeniability of other signatures.

4.4 Security

We will begin by discussing the possibility of creating false signatures. Since the verify and deny protocols are zero-knowledge, the strongest attack in this context is the one in which the enemy can ask for signatures on messages of his choice, and later tries to "sign" a different message. For our scheme, this means that the enemy can get triples of the form (T, r, s), such that

$$T^{Tzm} = y^r r^s,$$

where m is chosen by the enemy, and T is chosen by the signer. In the following, we make the worst case assumption that the enemy knows t, the discrete log of T, and z. This means that the enemy gets El Gamal signatures on numbers of the form $Ttmz$.

The only known consequence for the El Gamal scheme under this attack is that the enemy can construct new "signed" messages from the ones he is given (see [EG85]), but these new messages result from applying a hard-to-invert transformation to the

known messages, and therefore they cannot be controlled easily.[3] In the El Gamal scheme, as in ours, this problem is solved by applying a one-way hash function to the messages before signing.

Now suppose that the enemy is somehow able to create a signature (T, r, s) in our scheme on a message m. There are two cases:

1. If he also knows t, such that $T = \alpha^t$, he has created an El Gamal signature on $Ttmz$. So either he has found a completely new way to break the El Gamal system, or he has found a way to write the result of the hard-to-invert transformation mentioned above in the form $Ttmz$. We conjecture that this is a hard problem: m is in fact shorthand for a one-way hashed image of the actual message, the mapping $f(t) = \alpha^t t$ can reasonably be assumed to be one-way, and z is a constant. Thus none of the factors in $(Tt)mz$ can be easily controlled by the enemy.
2. The only remaining possibility is that the enemy can find values satisfying $(T^T)^{mz} = y^r r^s$ *without* knowing the discrete log of T. This problem could only be easier than case 1 if T is computed in some clever way from signatures the enemy got earlier from the signer. As discussed above, these are equivalent to El Gamal signatures on messages of a special form. But these messages cannot be controlled by the enemy, since they all involve a factor of the form $t\alpha^t$, where t is chosen independently by the signer. Hence it is reasonable to claim that these signed messages will be of no more use to the enemy than the ones he can construct himself from scratch. Under this assumption the enemy might as well choose T's for which he knows the discrete log, and hence we are back in case 1.

We now turn to the problem of verifying signatures without the aid of the signer. Again, since the verify and deny protocols are zero-knowledge, we can simply assume that the enemy is given some number of signatures he knows are valid, and is trying to guess whether a given triple (T, r, s) matches a message m, i.e. whether these values satisfy the usual equation $T^{Tmz} = y^r r^s$. The natural way to verify an equation like this is to compute each side and compare. But if the enemy can compute the left side, he can also compute $\alpha^{tz} = (T^{Tmz})^{(Tm)^{-1}}$. This computation of α^{tz} from α^z, which is known from the public key, and $T = \alpha^t$ is the Diffie-Hellman key exchange problem, which we assume is intractable.

When a number of valid signatures (T_i, r_i, s_i) are known to the enemy, we have a situation where $\alpha^{t_i z}, t_i, \alpha^z$ is known for a number of independently chosen t_i's (we make the worst case assumption that the signer releases t_i for each i). The enemy is trying to guess the value of α^{tz} for a t independent of all t_i's. For each i, the values mentioned so far could easily be generated with the same distribution by the enemy himself. The only additional information he has is that each $\alpha^{t_i z}$ can be expressed in a special form, namely as $(y^{r_i} r_i^{s_i})^{(T_i m_i)^{-1}}$. Since this expression involves the independently chosen r_i's, we conjecture that this extra information does not help the enemy.

[3] It is also possible to construct "signed" - but still hard to control - messages from scratch. They have the form $m = \alpha^B y^C BC^{-1}$, where B, C can be arbitrary integers. The construction that uses already signed messages is similar, but more complicated.

The definition of undeniability in section 3.1 calls for a signature simulator. As a consequence of the above discussion, we conjecture that the signature simulator need only output three random numbers for the triple (T, r, s).

4.5 Generalizations

This method of constructing undeniable signatures can be used in any group. However, if the group in question ($<\alpha>$) is not of prime order, one has to apply the more general protocol presented in [CEvdG88] for proving equality between logarithms. The reason for this is that the proof of lemma 4.1 does not work if w (which the signer chooses) generates a small subgroup.

5 Conclusion

We have introduced the concept of convertible undeniable signatures in which release of a single bit string by the signer turns all of his signatures, which were originally undeniable signatures, into ordinary digital signatures. The existence of such schemes is implied by the existence of digital signature schemes which exist if and only if a one-way function exists. We have presented a very efficient selectively convertible undeniable signature scheme.

Acknowledgement

The authors are thankful to Whitfield Diffie for suggesting a problem which motivated the idea of converting undeniable signatures to ordinary digital signatures.

References

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.

[BDLP89] Jørgen Brandt, Ivan Damgård, Peter Landrock, and Torben Pedersen. Zero-knowledge authentication scheme with secret key exchange. Submitted to Journal of Cryptology, 1989.

[CEvdG88] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology - proceedings of EUROCRYPT 87*, Lecture Notes in Computer Science. Springer-Verlag, 1988.

[Cha90] David Chaum. Zero-knowledge undeniable signatures. To appear in the proceedings of EUROCRYPT'90, 1990.

[CvA90] David Chaum and Hans van Antwerpen. Undeniable signatures. In *Advances in Cryptology - proceedings of CRYPTO 89*, Lecture Notes in Computer Science. Springer Verlag, 1990.

[EG85] Taher El Gamal. A public key cryptosystem and a signatures scheme based on discrete logarithms. In *Advances in Cryptology - proceedings of CRYPTO 84*, Lecture Notes in Computer Science. Springer-Verlag, 1985.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. In *Proceedings of the 25th IEEE Symposium on the Foundations of Computer Science*, 1984.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing*, 1989.

[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.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th IEEE Symposium on the Foundations of Computer Science*, pages 174–187, 1986.

[IL89] Russel Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, 1989.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing*, 1989.

[IY88] Russel Impagliazzo and Moti Yung. Direct minimum-knowledge computations. In *Advances in Cryptology - proceedings of CRYPTO 87*, Lecture Notes in Computer Science, pages 40–51. Springer-Verlag, 1988.

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

[Nao90] Moni Naor. Bit commitment using randomness. In *Advances in Cryptology - proceedings of CRYPTO 89*, Lecture Notes in Computer Science, pages 128 –136, 1990.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, 1990.