RSA-Based Undeniable Signatures*

Rosario Gennaro[†], Hugo Krawczyk[‡] and Tal Rabin[§]

Abstract

We present the first undeniable signatures scheme based on RSA. Since their introduction in 1989 a significant amount of work has been devoted to the investigation of undeniable signatures. So far, this work has been based on discrete log systems. In contrast, our scheme uses regular RSA signatures to generate undeniable signatures. In this new setting, both the signature and verification exponents of RSA are kept secret by the signer, while the public key consists of a composite modulus and a sample RSA signature on a single public message.

Our scheme possesses several attractive properties. First of all, provable security, as forging the undeniable signatures is as hard as forging regular RSA signatures. Second, both the confirmation and denial protocols are zero-knowledge. In addition, these protocols are efficient (particularly, the confirmation protocol involves only two rounds of communication and a small number of exponentiations). Furthermore the RSA-based structure of our scheme provides with simple and elegant solutions to add several of the more advanced properties of undeniable signatures found in the literature, including convertibility of the undeniable signatures (into publicly verifiable ones), the possibility to delegate the ability to confirm and deny signatures to a third party without giving up the power to sign, and the existence of distributed (threshold) versions of the signing and confirmation operations.

Due to the above properties and the fact that our undeniable signatures are identical in form to *standard* RSA signatures, the scheme we present becomes a very attractive candidate for practical implementations.

1 Introduction

The central role of digital signatures in the commercial and legal aspects of the evolving electronic commerce world is well recognized. Digital signatures bind signers to the contents of the documents they sign. The ability for any third party to verify the validity of a signature is usually seen as the basis for the "non-repudiation" aspect of digital signatures, and their main source of attractiveness. However, this universal verifiability (or

*A preliminary version of this paper appeared in the proceedings of CRYPTO'97

[†] IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598, USA Email: rosario@watson.ibm.com.

[‡] IBM T.J. Watson Research Center and Department of Electrical Engineering, Technion, Haifa 32000, Israel. Email: hugo@ee.technion.ac.il.

[§] Contact author. IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598, USA Email: talr@watson.ibm.com.

self-authenticating) property of digital signatures is not always a desirable property. Such is the case of a signature binding parties to a confidential agreement, or of a signature on documents carrying private or personal information. In these cases limiting the ability of third parties to verify the validity of a signature is an important goal. However, if we limit the verification to such an extent that it cannot be verified by, say, a judge in case of a dispute then the whole value of such signatures is seriously questioned. Thus, the question is how to generate signatures which limit the verification capabilities yet without giving up on the central property of non-repudiation.

An answer to this problem was provided by Chaum and van Antwerpen [CA89] who introduced *undeniable signatures*. Such signatures are characterized by the property that verification can only be achieved by interacting with the legitimate signer (through a *confirmation protocol*). On the other hand, the signer can prove that a forgery is such by engaging in a *denial protocol*. It is required that the following property be satisfied: if on a specific message and signature the confirmation protocol outputs that the pair is a valid signature then on the same input the denial protocol would not output that it is a forgery. The combination of these two protocols, confirmation and denial, protects both the recipient of the signature and the signer, and preserves the non-repudiation property found in traditional digital signatures. The recipient is protected since the ability of a signer to confirm a signature means that at no later point will the signer be able to deny the signature. For example, in the case of an eventual dispute, the recipient of the signature can resort to a designated authority (e.g., a judge) in order to demonstrate the signature's validity. In this case the signer will be required to confirm or deny the signature. If the signer does not succeed in denying (in particular, if it refuses to cooperate) then the signer remains legally bound to the signature (such will be the case if the alleged signature was a correct one). On the other hand the signer is protected by the fact that his signatures cannot be verified by unauthorized third parties without his own cooperation and the denial protocol protects him from false claims.

The protection of signatures from universal verifiability is not only justified by confidentiality and privacy concerns but it also opens a wide range of applications where verifying a signature is a valuable operation by itself. A typical example presented in the undeniable signatures literature is the case of a software company (or for this matter any other form of electronic publisher) that uses signature confirmation as a means to provides a proof of authenticity of their software to authorized (e.g., paying) customers only. This example illustrates the core observation on which the notion of undeniable signatures stands: *verification of signatures, and not only their generation, is a valuable resource to be protected.*

1.1 Components and Security of Undeniable Signatures Schemes

There are three main components to undeniable signature schemes. The signature generation algorithm (including the details of private and public information), the confirmation protocol, and the denial protocol. Signature generation is much like a regular signature generation, namely, an operation performed by the signer on the message which results in a string that is provided to the requester of the signature. The confirmation protocol is usually modeled after an interactive proof where the signer acts as the prover and the holder of the signature as the verifier. The input to the protocol is a message and its alleged signature (as well as the public key information associated with the signer). In case that the input pair is formed by a message and its legitimate signature then the prover can convince the verifier that this is the case, while if the signature does not correspond to the message then the probability of the prover to convince the verifier is negligible. Similarly, the denial protocol is an interactive proof designed to prove that a given input pair does *not* correspond to a message and its signature. However, if the alleged input signature does correspond to the input message then the probability of the prover to convince the verifier of the contrary is negligible. Note, that engaging in the confirmation protocol and having it fail is not an indication that the signature is invalid, this can only be established through the denial protocol. That is the confirmation protocol only establishes validity, and the denial – invalidity.

In addition to the above properties required from the confirmation and denial protocol, there are two basic security requirements on undeniable signatures. The first is unforgeability, namely, without access to the private key of the signer no one should be able to produce legitimate signatures by himself. This is similar to the unforgeability requirement in the case of regular digital signatures, but here the modeling of the attacker is somewhat more complex. In addition to having access to chosen messages signed by the legitimate signer, the attacker may also get to interact with the signer on different instances of the above confirmation and denial protocols, possibly on input pairs of his own choice. The second requirement is non-transferability of the signature, namely, no attacker (under the above model) should be able to convince any other party, without the cooperation of the legitimate signer, of the validity or invalidity of a given message and signature. Both of these requirements induce necessary properties on the components of an undeniable signature scheme. In particular, the confirmation and denial protocols should not leak any information that can be used by an attacker to forge or transfer a signature. As a consequence it is desirable that these protocols be zero-knowledge[1]. As for the strings representing signatures, they should provide no information that could help a party to get convinced of the validity (or invalidity) of the signature. Somewhat more formally, it is required that the legitimate signature(s) corresponding to a given message be *simulatable*, namely, they should be indistinguishable from strings that can be efficiently generated without knowledge of the secret signing key.

1.2 Advanced Properties of Undeniable Signatures

Much of the work on undeniable signatures has been motivated by the search for schemes that provide all of the above properties but that, in addition, enjoy some additional attractive properties. These include *convertibility* (the possibility to transform undeniable signatures into regular, i.e. self-authenticating, signatures by just publishing a short piece of information, [BCDP90]), *delegation* (enabling selected third parties to confirm/deny signatures but not to sign), *distribution of power* (threshold version of the signature and confirmation protocols, [Ped91]), *designated confirmer* schemes (in which the recipient of the signature is assured that a specific third party will be able to confirm the signature at a later time, [Cha94]), and *designated verifier* schemes (in which the prover can make sure

[1] At the minimum, if not zero-knowledge, these protocols should be proven to provide no "useful" information for the attacker to break the security of the scheme.

that only a specified verifier benefits from interacting with the prover on the confirmation of a signature, [JSI96]). More details on these extensions are provided in Section 5.

1.3 Previous Work on Undeniable Signatures

Since their introduction in 1989, undeniable signatures have received a significant attention in the cryptographic research community [CA89, Cha90, BCDP90, DY91, FOO91, Ped91, CvHP91, Cha94, Jak94, Oka94, Mic96, DP96, JSI96, JY96]. These works have provided a variety of different schemes for undeniable signatures with variable degrees of security, provability, and additional features. Interestingly, all these works are discrete logarithm based. In [BCDP90] the problem of constructing schemes based on different assumptions, in particular RSA, was suggested as a possible research direction.

Most influential are the works of Chaum and van Antwerpen [CA89] and Chaum [Cha90]. The first work introduces the notion of undeniable signatures and provides protocols which are the basis for many of the subsequent works. The second improves significantly on the initial solution by providing zero-knowledge versions of these protocols. The formalization of the basic notions behind undeniable signatures was mainly carried out in the works by Boyar, Chaum, Damgard and Pedersen [BCDP90] and by Damgard and Pedersen [DP96]. In [BCDP90] the notion of *convertible* schemes was introduced. In such schemes the signer can publish a short string that converts the scheme into a regular signature scheme. However the scheme presented in [BCDP90] was recently broken in [Mic96]. The repaired solution presented therein however does not come with a proof of security. [DP96] present the first convertible schemes with proven security (based on cryptographic assumptions).

1.4 Our Contribution

Our work is the first to present undeniable schemes based on RSA[2]. Our undeniable signature scheme produces signatures that are *identical in form* to RSA signatures. The essential difference from traditional RSA signatures is that in our case both the signature and verification exponents of RSA are kept secret by the signer, while the public key consists of a composite modulus and a sample RSA signature on a single public message.

Not only does our solution expand the list of available number-theoretic assumptions that suffice to build undeniable signatures, but it achieves and improves, as we show below, in a simple and elegant way several of the desirable properties of undeniable signatures.

Unforgeability: Our construction allows us to prove in a simple way that security of these signatures against forging is equivalent to the unforgeability of RSA signatures[3]. Provable unforgeability of undeniable signatures was presented for the first time in the recent paper by [DP96] where forgery of the proposed scheme is proven equivalent to forgery of the ElGamal scheme.

Simulatability: Non-transferability of an RSA signature is a non-standard requirement in the context of traditional RSA. We prove this property under the assumption that deciding

[2] Chaum in [Cha94] uses RSA signatures *on top* of regular undeniable signatures to provide "designated confirmer signatures"; however the underlying undeniable signatures are still discrete log-based.

[3] As with regular RSA, the use of a strong one-way hash function is assumed to provide unforgeability against chosen message attacks.

on the equality of discrete logarithms under different bases is intractable. A similar assumption is required in previous works as well[4] although, by itself, it is not always sufficient to prove simulatability of the undeniable signatures. For example in [DP96] the simulatability property is only conjectured to follow from such assumptions.

Zero-Knowledge: Our confirmation and denial protocols have the interactive proof properties as explained above and are also zero-knowledge. Therefore they do not leak any information that could otherwise be used for forging signatures. The soundness of our protocols (i.e. the guarantee that the prover/signer cannot cheat) relies on the use of composite numbers of a special form (specifically, with "safe prime" factors), which are secure moduli for RSA. A signer who chooses a modulus of a different form may have some way to cheat in our protocols. To force the signer to choose a "proper" modulus we require that he prove the correct choice of primes at the time he registers his public key with a certification authority. A discussion of this issue is presented in Section 4. An interesting question is whether our solution, or a different one, can work with a different kind of RSA moduli.

Efficiency: Our protocols are efficient (comparable to the most efficient alternatives found in the undeniable signatures literature). The confirmation protocol takes two rounds of communication (which is minimal for zero-knowledge protocols [GK96]) and involves a small number of exponentiations. The denial protocol is somewhat more expensive as it consists of a basic two-round protocol with small, but not negligible, probability of error (e.g., 1/1000) which needs to be repeated sequentially in order to further reduce the error probability. Its performance is still significantly better (by a factor of 10) than alternative protocols that only achieve probability 1/2 in each execution. We also note that in typical uses of undeniable signature schemes one expects to apply more frequently confirmation than denial. The latter is mainly needed to settle legal disputes.

Advanced Properties: In addition to the above security and efficiency properties, our solution naturally achieves several of the advanced features of undeniable signatures mentioned above. Once again it is the structure of RSA, in particular the presence of a secret verification exponent, that allows to achieve such properties very elegantly. Convertibility is achieved by publishing the verification exponent, thus converting the signatures into regular RSA signatures; delegation is achieved by providing the verification exponent to the delegated party which can then run the confirmation and denial protocols but cannot sign messages or forge signatures; distribution of the signature operation builds on the existing threshold solutions for RSA signatures; distribution of confirmation can be also achieved by an adaptation of the regular threshold RSA solutions. We can also adapt existing techniques for the construction of *designated confirmer* and *designated verifier* undeniable signatures, thus obtaining these variants also for our scheme. More details are provided in Section 5.

Standard RSA compatibility: An important practical advantage of our RSA-based undeniable scheme is that the signatures themselves are identical in form to standard RSA

[4] In our case the discrete logarithms are computed modulo a composite number while in previous works they are modulo a prime. In both cases, the problem is related to the problem of computing discrete logarithms which is considered to be hard (in the case of a composite modulus that difficulty is implied by the hardness of factoring and also directly by the assumed security of RSA). However, while the feasibility of computing discrete logarithm implies the feasibility of the above decision problem, the reverse direction is not known to hold.

signatures. In particular, this means that they fit directly into existing standardized communication protocols that use (regular) RSA signatures.

Technically, our work builds on previous ideas and protocols which we adapt to the RSA case. These previous solutions are designed to exploit the algebraic properties of cyclic groups like Z_p^* (and its subgroups). This is probably the main reason that subsequent work concentrated on these structures as well. Here we show that many of these ideas can be used in the context of RSA, thus answering in the affirmative a question suggested in [BCDP90]. In doing so we use ideas from the work of Gennaro et al. [GJKR96].

The paper is organized as follows: in Section 2 we give notation and some number theoretical lemmas. In Sections 3 and 4 we describe the new undeniable signature scheme and prove its properties and security. Section 5 includes extension of the scheme to variations of undeniable signatures suggested in the literature.

2 Preliminaries

Notation. Throughout the paper we use the following notation:
For a positive integer k we denote $[k] \stackrel{\text{def}}{=} \{1, \cdots, k\}$. Z_n^* denotes the multiplicative group of integers modulo n, and $\phi(n) = (p-1)(q-1)$ the order of this group. For an element $w \in Z_n^*$ we denote by $ord(w)$ the order of w in Z_n^*. The subgroup generated by an element $w \in Z_n^*$ is denoted by $<w>$.

The following technical lemmas are needed in our proofs in Section 3.

Lemma 1 *Let $n = pq$, where $p < q$, $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are all prime numbers. Then,*

1. *The order of elements in Z_n^* is one of the set $\{1, 2, p', q', 2p', 2q', p'q', 2p'q'\}$.*
2. *Given an element $w \in Z_n^* \setminus \{-1, 1\}$, such that $ord(w) < p'q'$ then either $gcd(w-1, n)$ or $gcd(w+1, n)$ is a prime factor of n.*

Proof. 1. To find the order of elements in Z_n^* it is enough to note that the maximal order of such an element is $2p'q'$ and that all the other orders must divide this one.
2. From the above property we get that if $1 < ord(w) < p'q'$, then $ord(w) \in \{2, p', q', 2p', 2q'\}$. If $ord(w) = 2$, $w \neq -1$, then $n|(w-1)(w+1)$ and then $gcd(w-1, n)$ must be a non-trivial factor of n. In case that $ord(w) = p'$, $w^{p'} \equiv 1 \bmod n \Rightarrow w^{p'} \equiv 1 \bmod q$. If $w \equiv 1 \bmod q$ then $w - 1$ is a multiple of q which is smaller than n, otherwise $p'|\phi(q) = 2q'$, a contradiction. A similar argument holds for $ord(w) = q'$. Finally in the case that $ord(w) = 2p'$, $w^{2p'} \equiv 1 \bmod n \Rightarrow (w^2)^{p'} \equiv 1 \bmod q$. If $w^2 = 1 \bmod q$ then either $w - 1$ or $w + 1$ is a multiple of q which is smaller than n, otherwise $p'|\phi(q) = 2q'$, a contradiction. Again a similar argument holds for $ord(w) = 2q'$. ■

As a consequence of the above lemma we can assume in our protocols that any value found by a party that does not know (and cannot compute) the factorization of n must be of order at least $p'q'$ in Z_n^* (except for 1,-1).

Lemma 2 *Let n be as in Lemma 1. Given an element w such that $ord(w) \in \{p'q', 2p'q'\}$ then for every $m \in Z_n^*$ it holds that $m^4 \in <w>$.*

Proof. We shall give the proof for the case $ord(w) = 2p'q'$ and show that $m^2 \in <w>$. If $m \in <w>$ then clearly the claim holds. Otherwise, $Z_n^* = <w> \cup m <w>$. If $m^2 \in <w>$ then we are done, otherwise it must hold that $m^2 \in m <w>$. This in return requires that $m \in <w>$, contradiction. The case of $ord(w) = p'q'$ is proved similarly. ■

3 The New Undeniable Signature Scheme

In this section we present the details of our scheme. We start by defining the following set:

$$\begin{aligned} \mathcal{N} &= \{n \mid n = pq, \;\; p < q, \;\; p = 2p' + 1, \;\; q = 2q' + 1, \\ &\quad \text{and } p, q, p', q' \text{ are all prime numbers}\} \end{aligned}$$

The system is set up by the signer in the following manner: chooses a random element $n \in \mathcal{N}$; selects elements $e, d \in \phi(n)$ such that $ed \equiv 1 \bmod \phi(n)$; chooses a pair (w, S_w) with $w \in Z_n^*$, $w \neq 1$, $S_w = w^d \bmod n$; sets the public key parameters to the tuple (n, w, S_w); sets the private key to (e, d).

We shall denote by $\mathcal{PK}$ the set of all tuples (n, w, S_w) generated as above. We refer the reader to Section 4.3 for a discussion on the form of the public key and how to verify its correctness. In particular, it is shown there that the value of w can always be set to a fixed number, e.g. $w = 2$. This simplifies the public key system and adds to the efficiency of computing exponentiations with base w.

3.1 Generating a Signature

To generate a signature on a message m the signer carries out a regular RSA signing operation, i.e. he computes $S_m = m^d \bmod n$, outputting the pair (m, S_m). More precisely, the message m is first processed through a suitable encoding (e.g., via one-way hashing) before applying the exponentiation such that the resultant signature scheme can be assumed to be unforgeable even against chosen message attacks (plain RSA does not have this property). Given a message m we will denote by $\bar{m}$ the output of such an encoding of m (we do not specify any encoding in particular)[5]. Thus, the resultant signature of m will be $S_m \stackrel{\text{def}}{=} \bar{m}^d \bmod n$. In the case of the pair (w, S_w) we will slightly abuse the notation and write S_w to denote $w^d \bmod n$ (i.e., we directly exponentiate w rather than $\bar{w}$).

3.2 Confirmation Protocol

In Figure 1 we present a protocol for confirming a signature. It is carried out by two players a prover and a verifier. The public input to the protocol are the public key parameters, namely $(n, w, S_w) \in \mathcal{PK}$, and a pair $(m, \hat{S}_m)$. For the case that $\hat{S}_m$ is a valid signature of m, then P will be able to convince V of this fact, while if the signature is invalid then no prover (even a computationally unbounded one) will be able to convince V to the contrary except for a negligible probability.

[5]For simplicity we will assume a deterministic encoding; however randomized encodings, e.g. [BR96], can be used as well but then, in our case, the random bits used for the encoding need to be attached to the signature.

This protocol is basically the same as the protocol of Gennaro et al. [GJKR96] (based on [Cha90]) where it is used in a different application, namely, threshold RSA. Our variation on this protocol uses the verification key e rather than the signature key d as originally used in [GJKR96] (in their case, the signer knows only d but not e). Still the basic proof given in that paper applies to our case due to the symmetry that exists between d and e when both exponents are kept secret. This modification allows us to provide solutions where the ability to confirm signatures can be delegated to third parties while keeping the ability to sign new messages only for the original signer (it also allows for a distributed prover solution). See Section 5 for the details.

The idea of the protocol is for the verifier to test the alleged signature on m by producing a related element which looks random to the signer and for which the verifier knows the signature (given that the signature on m is correct). This "blinded" element is created via the exponentiation of the message m with a random exponent i and its multiplication with a random exponent j of the value w (for which the correct signature S_w is publicly known). Intuitively, a cheating prover needs to find the values of i and j in order to cheat. However, there are many pairs of exponents that give the same result and we show that the prover (even if computationally unbounded) cannot distinguish among them.

An interesting aspect of this protocol is that a prover could succeed in convincing the verifier to accept a signature on m even when this signature is not $\bar{m}^d \bmod n$ but $\alpha\bar{m}^d \bmod n$ where α is an element of order 2 (in Z_n^*). [GJKR96] solve this problem through the assumption (valid in their case) that the prover cannot factor n and thus cannot find such an element α. In our case, this assumption does not hold. We deal with this problem by accepting as valid signatures also these particular multiples of $\bar{m}^d$. On the other hand, when designing the denial protocol we make sure that the signer cannot deny a signature of this extended form. That is, we define the set of valid signatures for a message m as $\mathcal{SIG}(m) \stackrel{\text{def}}{=} \{S_m : S_m = \alpha\bar{m}^d,\ \ ord(\alpha) \le 2\}$.

Signature Confirmation Protocol

Input: Prover: Secret key $(d, e) \in [\phi(n)]^2$
Common: Public key $(n, w, S_w) \in \mathcal{PK}$,
$m \in Z_n^*$ and alleged $\hat{S}_m$

1. V chooses $i, j \in_R [n]$ and computes $Q \stackrel{\text{def}}{=} \hat{S}_m^{2i} {S_w}^j \bmod n$
   $V \longrightarrow P : Q$
2. P computes $A \stackrel{\text{def}}{=} Q^e \bmod n$
   $P \longrightarrow V : A$
3. V verifies that $A = \bar{m}^{2i} w^j \bmod n$.
   If equality holds then V accepts $\hat{S}_m$ as the signature on m, otherwise "undetermined".

Figure 1: Proving that $\hat{S}_m \in \mathcal{SIG}(m)$ (ZK steps omitted)

For ease of exposition the protocol in Figure 1 appears in a non zero-knowledge format. However, there are well-known techniques [GMW91, BCC88, Gol95] to add the zero-knowledge property to the above protocol using the notion of a *commitment function*: Instead of P sending A in Step 2, he sends a commitment $commit(A)$, after which V reveals to P the values of i and j. After checking that $Q \stackrel{\text{def}}{=} \hat{S}_m^{2i} {S_w}^j \bmod n$, P sends A to V . The verifier checks that A corresponds to the value committed by P and then performs the test of Step 3 above.

The zero-knowledge condition is achieved through the properties of the commitment function, namely, (i) $commit(x)$ reveals no information on x, and (ii) P cannot find x' such that $commit(x) = commit(x')$. Commitment functions can be implemented in many ways. For example, in the above protocol $commit(A)$ can be implemented as a probabilistic (semantically secure) RSA encryption of A using a public key for which the private key is not known to V (and possibly, not even known to P). To open the commitment, P reveals both A and the string r used for the probabilistic encryption. This implementation of a commitment function is very efficient as it does not involve long exponentiations (and is secure since we assume our adversary, the verifier in this case, is unable to break RSA).

Theorem 1 Confirmation Theorem. *Let* $(n, w, S_w) \in \mathcal{PK}$.

Completeness. *Given* $S_m \in \mathcal{SIG}(m)$, *if* P *and* V *follow the Signature Confirmation protocol then* V *always accepts* S_m *as a valid signature.*

Soundness. *A cheating prover* P^*, *even computationally unbounded, cannot convince* V *to accept* $\hat{S}_m \notin \mathcal{SIG}(m)$ *with probability greater than* $\frac{O(1)}{p'}$.

Zero-knowledge. *The protocol is zero-knowledge, namely, on input a message and its valid signature, any (possibly cheating) verifier* V^* *interacting with prover* P *does not learn any information aside from the validity of the signature.*

Proof.

Completeness. Immediate from inspection of the protocol. Note that raising $\hat{S}_m$ to an even power eliminates any extra factor of order 2, if such exists, from the signature (such factors are allowed by definition of $\mathcal{SIG}(m)$).

Soundness. We adapt the proof from [GJKR96] to our case. The prover's probability to cheat, i.e. to convince V to accept $\hat{S}_m \notin \mathcal{SIG}(m)$, is maximized by choosing A that passes V's test (in Step 3) with maximal probability (relative to the values i, j chosen by V). As the prover chooses A after having seen the "challenge" Q from V (and based on its knowledge of $\hat{S}_m, m, w, d, e$ and n), the proof of soundness needs to capture that some information on i, j (at least from the information theoretic point of view) is available to the prover when selecting A.

In the actual protocol, V chooses i, j randomly from the set $[n]$; for simplicity of analysis we will assume that these values are chosen from $[\phi(n)]$, and will account for the event that either i or j fall outside of this range in the prover's probability to cheat. The probability of such event (i.e., that i or $j \notin [\phi(n)]$), denoted by π_1, is at most $2\frac{n-\phi(n)}{n}$. Thus, in the sequel, we assume $i, j \in_R [\phi(n)]$.

We define $I(Q) = \{i \in [\phi(n)] : \exists j, Q = \hat{S}_m^{2i} S_w^j \bmod n\}$. Since $\hat{S}_m \notin \mathcal{SIG}(m)$ we can write $\hat{S}_m = \alpha \bar{m}^d$, for $\alpha \in Z_n^*$, $ord(\alpha) > 2$. In Step 3 the verifier will check whether

$$A = \bar{m}^{2i} w^j = \alpha^{-2ei} \hat{S}_m^{2ei} S_w^{ej} = \alpha^{-2ei} Q^e \tag{1}$$

As the value α has been set in advance, then for any A the number of i's which satisfy Equation (1) is the same as the number of i's such that $\alpha^{2i} = A^{-d}Q$ which is at most $\phi(n)/ord(\alpha)$. Given Q, V's choice of i is uniformly distributed over $I(Q)$, as for each $i \in I(Q)$ there is the same number of values j which satisfy the equation $Q = \hat{S}_m^{2i} S_w^j \bmod n$. Thus, the probability of P to succeed is at most $\frac{\phi(n)/ord(\alpha)}{|I(Q)|}$. We denote the later quantity by π_2 and proceed to bound it by bounding $|I(Q)|$.
Clearly if V follows the protocol then $I(Q)$ is not empty. Now we show that $\forall Q$ properly formed, $|I(Q)| \geq ord(w)$.

If $I(Q)$ is non-empty then for a value $i \in I(Q)$ and Δ such that $\hat{S}_m^{2\Delta} \in <S_w>$, it holds that $i + \Delta \in I(Q)$ (because there exist j, j' such that $Q = \hat{S}_m^{2i} S_w^j$ and $\hat{S}_m^{2\Delta} = S_w^{j'}$ from which it follows that $Q = \hat{S}_m^{2(i+\Delta)} S_w^{j-j'}$). Therefore, we get that $\{i + \Delta : \hat{S}_m^{2\Delta} \in <S_w> \text{ and } \Delta < \phi(n)\} \subseteq I(Q)$. Thus, the size of $I(Q)$ is at least the size of the set $D = \{\Delta < \phi(n) : \hat{S}_m^{2\Delta} \in <S_w>\}$. We proceed to bound the size of D. Using standard arguments it is easy to show that if δ is the minimal non-zero element of D then the elements of D are exactly the multiples of δ (smaller than $\phi(n)$). Thus, $|D| = \phi(n)/\delta$. We now show that $\delta \leq \frac{\phi(n)}{ord(w)}$. Let $i_1 < i_2 \leq \delta$. The cosets $\hat{S}_m^{2i_1} <S_w>$ and $\hat{S}_m^{2i_2} <S_w>$ are disjoint (a common element would imply that $\hat{S}_m^{2(i_2-i_1)} \in <S_w>$ in contradiction to the minimality of δ). Thus, $\hat{S}_m^2 <S_w>$, $\hat{S}_m^4 <S_w>$, $\cdots \hat{S}_m^{2\delta} <S_w>$, are δ disjoint cosets in Z_n^* each of size $|<S_w>|$. The latter size is exactly $ord(w)$ since $<S_w>=<w>$, as $S_w = w^d$ and d is relatively prime to $\phi(n)$. We thus have $\delta \leq \frac{|Z_n^*|}{ord(w)} = \frac{\phi(n)}{ord(w)}$. In conclusion, $|I(Q)| \geq |D| = \phi(n)/\delta \geq ord(w)$. Combining all the above we get that $\pi_2 < \frac{\phi(n)}{ord(\alpha) ord(w)}$, and the total failure probability is at most $\pi_1 + \pi_2$.

(We stress that the above holds also for a computationally unbounded cheating prover, and that the bound is tight for such a prover, up to the term $\pi_1 = 2^{\frac{n-\phi(n)}{n}}$.)

The above bound on the probability of success of a cheating prover is given in terms of the order of elements in the group Z_n^*. Recall that we are using n's of a special form, i.e. $n = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$, with p, q, p', q' all large primes. Assume w.l.o.g. that $p' < q'$. Using Lemma 1 we can claim that $ord(w) \geq p'q'$ and $ord(\alpha) \geq p'$, thus $\pi_2 < \frac{4}{p'}$. Also, the expression $2^{\frac{n-\phi(n)}{n}}$ is at most $2/p'$ in this case. This proves the soundness statement in the theorem.

Zero-Knowledge. Immediate (see remarks after the description of the protocol). ■

3.3 Denial Protocol

Figure 2 exhibits the Denial Protocol. The public input to the protocol are the public key parameters, namely $(n, w, S_w) \in \mathcal{PK}$, and a pair $(m, \hat{S}_m)$. In the case that $\hat{S}_m \notin \mathcal{SIG}(m)$, then P will be able to convince V of this fact, while if $\hat{S}_m \in \mathcal{SIG}(m)$ then no prover (even a computationally unbounded one) will be able to convince V that the signature is invalid except with negligible probability.

Our solution is based on a protocol due to Chaum [Cha90], designed to prove in zero-knowledge the inequality of the discrete logarithms of two elements over a prime field Z_p

Denial Protocol

Input: Prover: Secret key $(d, e) \in [\phi(n)]^2$
Common: Public key $(n, w, S_w) \in \mathcal{PK}$,
$m \in Z_n^*$ and alleged non-signature $\hat{S}_m$

1. V chooses $i = 4b$, $b \in_R [k]$ and $j \in_R [n]$.
   Sets $Q_1 = \bar{m}^i w^j \bmod n$ and $Q_2 = \hat{S}_m^i {S_w}^j \bmod n$
   $V \longrightarrow P : (Q_1, Q_2)$
2. P computes $Q_1/Q_2^e = (\frac{\bar{m}}{\hat{S}_m^e})^i$ and computes $i = 4b$ by testing all possible values of $b \in [k]$.
   If such a value was found then P sets $A = i$, otherwise abort.
   $P \longrightarrow V : A$
3. V verifies that $A = i$. If equality holds then V rejects $\hat{S}_m$ as a signature of m, otherwise, undetermined.

Figure 2: Proving that $\hat{S}_m \notin \mathcal{SIG}(m)$ (ZK steps omitted)

relative to two different bases. The protocol and proof presented in the above paper do not work over Z_n^* for a composite n as required here, in particular, since they strongly rely on the existence of a generator for the multiplicative group Z_p^*. However, a careful adaptation of that protocol and a more involved proof can be shown to solve our problem over Z_n^*.

The protocol (see Figure 2) works in the following manner: the verifier gives the prover in Step 1 two values from which the prover can extract, using the verification exponent e, the quotient $(\frac{\bar{m}}{\hat{S}_m^e})^i$, for some value i chosen by V. The verifier accepts the run of the protocol only if the prover can find the value i. We will see that if $\hat{S}_m$ is *not* a valid signature of the message m, then P exhaustively searches the range for the desired value of i. However, in case that $\hat{S}_m$ is a valid signature of m, the above quotient equals 1 regardless of the value of i. Then the prover cannot learn any information about i and can only try to guess that value (see the proof below for a formal argument).

In order to allow for an exhaustive search of i by P , one needs to choose the range of i to be relatively small. If the upper bound on i is set to some value k, then the prover needs to perform k multiplications (of the value $\frac{\bar{m}}{\hat{S}_m^e}$) to find i. The protocol has thus probability of error $\frac{1}{k}$. Notice that by choosing $k = O(\log n)$ the cost of the exhaustive search is then roughly equivalent to a single long exponentiation. On the other hand, the probability of cheating in this case is $1/k$. If we take, for example, $k = 1024$ we can repeat the protocol ten times in order to achieve a security of $\frac{1}{2^{100}}$. As stated in the introduction this allows for a ten fold increase in efficency relative to alternative protocols that need to repeat a subprotocol that bounds the cheating probability by only $1/2$.

The protocol as presented in Figure 2 omits the steps that make it zero-knowledge. This is similar to the case of the confirmation protocol. Yet, in this protocol special care needs to be taken in Step 2. If the (honest) prover does not find a value i that satisfies the equation, which means that V is cheating, P aborts the execution of the protocol.

Though aborting the protocol does not reveal much information it does reveal some, and in the zero-knowledge version we do not want even this much information to leak. Thus, P should continue the execution of the protocol by committing to the value 0, in a "dummy commitment" this will conceal the information of whether a value i was found or not. Note that in the case where no i was found, the verifier will be exposed later as a cheater and the commitment of 0 will never be revealed.

Theorem 2 Denial Protocol *Let* $(n, w, S_w) \in \mathcal{PK}$.

Completeness. *Assuming that* $\hat{S}_m \notin \mathcal{SIG}(m)$, *and if* P *and* V *follow the protocol then* V *always accepts that* $\hat{S}_m$ *is not a valid signature of* m.

Soundness. *Assuming that* $\hat{S}_m \in \mathcal{SIG}(m)$ *then a cheating prover* P^*, *even computationally unbounded, cannot convince* V *to reject the signature with probability greater than* $\frac{1}{k} + \frac{O(1)}{p'}$.

Zero-knowledge. *The protocol is zero-knowledge, namely, on input a message and a non-valid signature, any (possibly cheating) verifier* V^* *interacting with prover* P *does not learn any information aside from the fact that* $\hat{S}_m$ *is in fact not a valid signature for the message* m.

Proof.

Completeness. In the following we omit the $\bmod n$ from the notation. We can assume that $\hat{S}_m = \alpha \bar{m}^d$ where $ord(\alpha) \geq p'$, this holds as $\hat{S}_m, \bar{m}^d$ are in Z_n^* and hence α exists furthermore $\hat{S}_m \notin \mathcal{SIG}(m)$ indicating that $ord(\alpha) \geq p'$. The prover will not be able to find the value i only if $ord(\frac{\bar{m}}{\hat{S}_m^e}) < 4k$. The order $ord(\frac{\bar{m}}{\hat{S}_m^e}) = ord(\frac{\bar{m}}{\alpha^e \bar{m}}) = ord(\alpha^e)$. As $(e, \phi(n)) = 1$ we have that $ord(\alpha^e) \geq p'$. As we take $k \ll p'$ we prove our claim.

Soundness. We stress that the following proof holds also for a computationally unbounded prover. In order for P to convince V that $\hat{S}_m$ is not a valid signature he must send V a value A such that $A = i$. As $\hat{S}_m \in \mathcal{SIG}(m)$ it holds that $\hat{S}_m = \alpha \bar{m}^d$ where $ord(\alpha) \leq 2$. Thus, $Q_2 = \hat{S}_m^i S_w^j = \alpha^i \bar{m}^{di} w^{dj} = (\bar{m}^i w^j)^d$. As $\bar{m}^4 \in <w>$ (Lemma 2), it holds that $\exists l$ such that $w^l = \bar{m}^4$. Thus, $Q_1 = \bar{m}^i w^j = w^{lb+j}$ and $Q_2 = (\bar{m}^i w^j)^d = w^{(lb+j)d}$. A computationally unbounded prover, can compute the value r such that $Q_1 = w^r = w^{lb+j}$. Then to compute i the prover still needs to find b, that is he needs to solve the equation $r = lb + j \bmod ord(w)$. Assuming that $j \in_R [\phi(n)]$ then for every possible value of b there would be $\frac{\phi(n)}{ord(w)}$ possible value of j indicating that the best P could do is to guess at random giving a probability of $\frac{1}{k}$. Allowing for the fact that $j \in_R [n]$ (instead of $j \in_R [\phi(n)]$ as assumed above) we get $\frac{1}{k} + \frac{O(1)}{p'}$.

Zero-knowledge. The protocol as presented in Figure 2 is *not* zero-knowledge. However, as explained above, using the same techniques described in the confirmation protocol (and a "dummy commitment" in case of early abortion) we achieve zero-knowledge for this protocol as well. ■

4 Security Analysis

We do not present here a formal treatment of the security requirements of undeniable signatures. For such a formal and complete treatment we refer the reader to the paper by Damgard and Pedersen [DP96]; an outline of these notions can be found above in our introduction (in particular, in Section 1.1). Here we argue the security properties of our solution based on this outline, and the zero-knowledge results from previous section.

4.1 Unforgeability of Signatures

In this section we will prove the following theorem.

Theorem 3 *Assuming that the underlying RSA signatures are unforgeable (against known and/or chosen message attacks) then our undeniable signatures are unforgeable (against the same attacks).*

As noted before, RSA is not directly immune against chosen message attacks but we assume this to be countered by additional means, e.g. by the appropriate encoding of the message prior to the exponentiation – see Section 3.1.

Assume that there exists a forger $\mathcal{F}$ which can forge an undeniable signature in our scheme after receiving the undeniable public key pair and interacting with the signer in confirmation and denial protocols. That is the forger outputs a pair (m, S_m) where $S_m = \alpha\bar{m}^d, \;\; ord(\alpha) \leq 2$. We will construct an attacker $\mathcal{A}$ who will use this forger and forge regular RSA signatures. Given the RSA public key (n, e) of a signer $\mathcal{S}$ for which $\mathcal{A}$ would like to forge a signature he proceeds as follows. He chooses a random value r and sets the public key of the undeniable signature scheme to the triple $(n, w = r^e \bmod n, S_w = r)$ and gives these values to $\mathcal{F}$. When $\mathcal{F}$ requests an undeniable signature on a message m the attacker $\mathcal{A}$ asks $\mathcal{S}$ to sign this message and hands $\mathcal{F}$ the pair (m, S_m). When $\mathcal{A}$ is requested by $\mathcal{F}$ to participate in a confirmation/denial protocol on a pair (m, S) then $\mathcal{A}$ checks if m is a previously signed message and $S_m = S$, if yes then he interacts with the forger in a confirmation protocol otherwise he interacts in a denial protocol. The attacker utilizes the simulator for these protocols. We assume that the pair (m, S) still hasn't helped the attacker to factor the modulus. After this procedure the forger $\mathcal{F}$ output a forgery of our undeniable scheme, i.e. a pair $(m, \bar{m}^d)$ or $(m, \alpha\bar{m}^d)$ where $ord(\alpha) = 2$. A forgery for the RSA scheme is achieved as follows. If the pair is $(m, \bar{m}^d)$ then $\mathcal{A}$ outputs this value directly, as it is a standard RSA signature. In the second case, $\mathcal{A}$ holds the value e and thus by computing $(\alpha\bar{m}^d)^e/\bar{m}$ $\mathcal{A}$ extracts α (note that e is odd) and in return factors n which enables to generate forgeries. Note that $\mathcal{A}$ has asked the signer only for signatures which the forger has asked, thus the forger's output must be of a signature on a message which was not previously signed by the signer of the standard RSA scheme.

4.2 Indistinguishability of Signatures

A basic goal of undeniable signatures is that no one should be able to verify the validity (or invalidity) of a message and its (alleged) signature without interacting with the legitimate signer in a confirmation (or denial) protocol. Following [DP96] we need to show that given the public key information and any message m (but not the signature exponent d) one can efficiently generate a *simulated signature* $s(m)$ of m, in the sense that the distribution of simulated signatures cannot be distinguished (efficiently) from the distribution of true signatures on m. We achieve this property in the following way. Given any message m, we apply to it the encoding $\bar{m}$ as determined by the underlying RSA scheme and then raise the result $\bar{m}$ to a random exponent modulo n (i.e., $s(m) = \bar{m}^r \bmod n$, for $r \in_R [n]$). Notice that distinguishing $s(m)$ from the signature $\bar{m}^d \bmod n$ on m is equivalent to deciding whether

$$\log_m(s(m)) \stackrel{?}{=} \log_w(S_w) \tag{2}$$

where the discrete logarithm operation is taken in Z_n^*. This problem has no known efficient solution, though its equivalence to RSA, factoring, or the discrete logarithm problems has not been established.[6] We thus require the following intractability assumption in order to claim the hardness of distinguishing between valid and simulated signatures.

Assumption EDL: For values $n, w, S_w, \bar{m}$, and $s(w)$ as defined above it is infeasible to decide the validity of equation (2) over Z_n^*.

Note that the encoding of m is part of the assumption. We stress that the analogous assumption modulo a prime number is necessary for claiming the security of previous undeniable signature schemes as well (see [DP96]). However, while we can prove that the EDL assumption implies the simulatability of our signatures, in [DP96] this implication is not proven but just conjectured to hold.

Theorem 4 *Under the above EDL assumption, our signatures are simulatable and hence cannot be verified without the signer's (or its delegated confirmers) cooperation.*

Remark: The above theorem does not concern itself with a general problem of undeniable signatures pointed out first by Desmedt and Yung [DY91]. It is possible that the signer is fooled into proving a signature to several (mutually distrustful) verifiers while he is convinced of proving the signature to only one of them. We will address this problem in Section 5.

4.3 Choosing the Signer's Keys

In Section 3 we defined what the public and private parameters for the signer should be. Our analysis of the (soundness of the) confirmation and denial protocols depends on these parameters being selected correctly. Typically, the verification of this public key will be done whenever the signer registers it with a trusted party (e.g., a certification authority). Here we outline protocols to check the right composition of the modulus n, the sample element w, and the fact that S_w is chosen as a power of w (the latter serves as the "commitment" of the signer to the signature exponent d). Notice that these protocols are executed *only once* at registration time and not during the subsequent signing/verification operations. We

[6]The problem is at least as hard as the decisional Diffie-Hellman problem (i.e., given a triple (g^x, g^y, r) decide whether $r = g^{xy}$). For the case of a composite modulus (our case), the related search problem (given g^x, g^y find g^{xy}) is known to be at least as hard as factoring [Shm85, McC88]. A similar result for the decisional problem is not known; such a result would imply that all the security aspects of our construction could be based solely on the security of RSA.

denote by V the entity that acts as the verifier of these parameters, and by P the signer that proves its correct choices.

VERIFICATION THAT w IS OF HIGH ORDER. Specifically, we use in our analysis the assumption that w is an element of order at least $p'q'$. By virtue of Lemma 1 all that V needs to verify is that $w \notin \{-1, 1\}$ and that $gcd(w-1, n)$ is not a factor of n. Actually, the value w can be chosen as a constant, e.g. $w = 2$, for *all* the undeniable signatures public keys. Such a value must always pass the verification (or otherwise factoring is trivial).

VERIFICATION THAT $S_w \in <w>$. The following protocol is essentially the protocol for proving possession of discrete logarithms as presented in [CEvdG87], once again modified in order to work with composite moduli. The signer P chooses a value $r \in_R [\phi(n)]$ and sends to V the value $w' = w^r$. The verifier V answers with a random bit b. If $b = 0$, P returns the value r, otherwise it returns the value $d + r \bmod \phi(n)$. In the first case, V checks whether $w^r = w'$, and in the second, whether $w^{(r+d)} = w' S_w$. If $w \notin <w>$ then the probability that P passes this test is $1/2$. By repeating this procedure k times the probability that the dealer can cheat reduces to 2^{-k}. The protocol is statistical zero-knowledge as the simulator does not know $\phi(n)$, but can use the uniform distribution on $[1..n]$ to statistically approximate the one on $[1..\phi(n)]$. As a practical matter, we observe that this protocol can be performed non-interactively if one assumes the existence of an ideal hash function (a la Fiat-Shamir [FS86]).

VERIFICATION OF THE PRIME FACTORS. We need to check that the signer chooses the modulus n of the right form, i.e. $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ and p, q, p', q' are all prime numbers. Recently, Gennaro, Micciancio and Rabin [GMR98] have presented a zero-knowledge proof to verify that a composite is of a slightly different form, where p, q are of the form $p = 2p_1^\alpha + 1$ and $q = 2q_1^\beta + 1$. Applying their techniques in our setting even though the test if for a prime power the result is that it equates the signer's probability of cheating with the probability of factoring his composite. See [GMR98] for details.

5 Extensions

Our protocols lend themselves to many of the existing extensions in the literature for undeniable signatures.

5.1 Convertible Undeniable Signatures

This variation appeared first in [BCDP90], and secure schemes based on ElGamal signatures have been recently presented in [DP96]. Convertible undeniable signatures enable the signer to publish a value which transforms the undeniable signature into a regular (i.e., self-authenticating) digital signature. In our scheme conversion can be easily achieved by simply publishing the value $e = d^{-1} \bmod \phi(n)$. Doing so the signer will transform the undeniable signatures into regular RSA signatures with public key (n, e). Notice that this will automatically imply the security (i.e., unforgeability) of the converted scheme, based on the security of regular RSA signatures.[7]

[7]Notice that this holds if the signer issued for the message m its *intended* signature $S_m = \bar{m}^d \bmod n$. If, instead, the signer generated a signature of the form $S_m = \alpha \bar{m}^d$, where α is an element of order 2, then when

Selective Conversion. In some applications it may be desirable to convert only a subset of the past signatures (*selective conversion* [BCDP90]). For this scenario we can make use of a non-interactive zero-knowledge confirmation proof for those messages.

Let $(m_1, S_1), \ldots, (m_\ell, S_\ell)$ be the message–signature pairs that the signer wants to convert. If the signer were allowed to interact with an honest verifier he could use the public-coin, statistical zero-knowledge, confirmation protocol in Figure 3. The protocol is based on a similar one in [CP92] which works for prime moduli. [8]

Honest Verifier Signature Confirmation Protocol

Input: Prover: secret $d, e \in [\phi(n)]$
Common: RSA composite $n \in \mathcal{N}$, sample message $w \in Z_n^*$, signature S_w, messages $m_1, \ldots, m_\ell$, claimed $S_1, \ldots, S_\ell$

1. P chooses $r \in_R [\phi(n)]$ and computes $\alpha_i \stackrel{\text{def}}{=} \bar{m}_i^r \bmod n$ for $i = 1, \ldots, \ell$ and $\beta \stackrel{\text{def}}{=} w^r \bmod n$
   $P \longrightarrow V : \alpha_1, \ldots, \alpha_\ell, \beta$
2. V chooses $c \in_R [n]$
   $V \longrightarrow P : c$
3. P computes $a = r + cd \bmod \phi(n)$
   $P \longrightarrow V : a$
4. V checks if:
   $\alpha_i S_i^c = \bar{m}_i^a \bmod n$ for $i = 1, \ldots, \ell$ and $\beta S_w^c = w^a \bmod n$
   If all equalities holds then V accepts the S_i's as the signatures on the m_i's, otherwise it rejects.

Figure 3: Proving that $S_i \in \mathcal{SIG}(m_i)$ to an honest verifier

In order to use this protocol for selective conversion we need to make it non–interactive using standard techniques (e.g. computing the challenge via a hash-function applied to the first message.) Security is retained in the random oracle model [BR93].

5.2 Delegation

The idea is for the signer to delegate the ability to confirm and deny to a third party without providing that party the capabilities to generate signatures. In the literature this notion is usually treated in the context of convertibility of signatures. However the two notions are conceptually different. Clearly the information used in order to delegate confirmation/denial authority to a third party if made public would basically convert undeniable signatures into

e is made public it is easy to recover α (and then the factorization of n) from a triple $(m, S_m = \alpha\bar{m}^d, e)$ since e is odd. We stress that although we consider as valid also signatures of that form (see Section 3.2), it is in the interest of the prover not to generate them in that way.

[8] We stress that we did not use this protocol as our main confirmation protocol since it is zero-knowledge only against an honest verifier.

universally verifiable ones. However the converse is not necessarily true. It may be that the information used to convert signatures, if given secretly to a third party, would still not allow that party to prove *in a non–transferable way* the validity/invalidity of a signature[9]. In our setting the signer can simply give the third party the key e which is the only needed information in order to carry out successfully the denial and confirmation protocols. Clearly, the recipient of e cannot sign by itself as this is the basic assumption behind regular RSA signatures.

5.3 Distributed Provers (and signers)

Distributed Provers for undeniable signatures were introduced by Pedersen [Ped91]. With distributed provers the signer can delegate the capability to confirm/deny signatures, without needing to trust a single party. This is obtained by sharing the key, used to verify signatures, using a (verifiable) secret sharing scheme among the provers. This way only if t out of the n provers cooperate it is possible to verify or deny a signature. The existing solutions for threshold RSA signatures [DDFY94, GJKR96] can then be used to obtain an efficient distributed scheme as the only operation needed during confirmation or denial protocols is RSA exponentiations. The fault-tolerance of the protocol in [GJKR96] guarantees the security of the scheme even in the presence of t (out of n) maliciously behaving provers.

As Pedersen pointed out in [Ped91], undeniable signatures with distributed provers present some difficulties. Indeed when the provers are presented with a message and its alleged signature, they have to decide which protocol (either the denial or the confirmation) to use. They can do this by first distributively checking for themselves if the claimed signature is correct or not. But this in turn means that a dishonest prover can use the other provers as an oracle to the verification key at his will. The problem applies to our schemes as well. Several ways of dealing with the problem have been suggested in the literature [Ped91, JY96] some of which easily extend to our scenario.

Also solutions for threshold RSA allow to share the power to sign (in addition to the power to verify/deny signatures) among several servers. Once again in case of possibly maliciously behaving signers a fault-tolerant scheme as [GJKR96] must be used.

5.4 Designated Verifier

The following problem of undeniable signatures has been pointed out (see [DY91, Jak94]): in general a mutually suspicious group of verifiers can get simultaneously convinced of the validity of a signature by interacting with the signer in a single execution of the confirmation protocol (in other words, the signer may believe that it is providing the signature confirmation to a single verifier while in actuality several of them are getting convinced at once). This is possible by having the "official" verifier act as the intermediary (or man in the middle) between the prover and the larger set of verifiers. While this is not always a problem, in some cases this may defeat the purpose of undeniable signatures (e.g., if the signer wants to receive payment from each verifier that gets a signature confirmation).

Jakobsson et al. [JSI96] present a solution to this problem through the notion of *designated verifiers proofs* that is readily applicable to our scheme. All that is required is for

[9] An example is the above scheme for the selective conversion of signatures.

the verifier to have a public key. Then when the prover commits to his answer during the zero-knowledge steps of our protocols he will use a trapdoor commitment scheme (as in [BCC88]) which the verifier can open in any way. This will prevent the verifier from "transferring" the proof (see [JSI96] for the details).

5.5 Designated Confirmer

Designated confirmer undeniable signatures were introduced by Chaum in [Cha94] and further studied by Okamoto in [Oka94]. This variant of undeniable signature is used to provide the recipient of a signature with a guarantee that a specified third party (called a "designated confirmer") will later be able to confirm that signature. Notice the difference between this variant and the delegation property described above. Indeed in the present case the signature is specifically bound at time of generation to a particular confirmer. The techniques of [Cha94, Oka94] easily extend to our scheme.

An Open Question

It would be interesting to see whether efficient undeniable signatures could be designed using more general form of composite.

Acknowledgments

We would like to thank Don Coppersmith and Ivan Damgård for useful suggestions.

References

[BCC88] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *JCSS*, 37(2):156–189, 1988.

[BCDP90] J. Boyar, D. Chaum, I. Damgård, and T. Pedersen. Convertible undeniable signatures. In A. J. Menezes and S. Vanstone, editors, *Advances in Cryptology — Crypto '90*, pages 189–205, Berlin, 1990. Springer-Verlag. Lecture Notes in Computer Science No. 537.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures, how to sign with RSA and Rabin. In Ueli Maurer, editor, *Advances in Cryptology — Eurocrypt '96*, pages 399–416, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science No. 1070.

[CA89] David Chaum and Hans Van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology — Crypto '89*, pages 212–217, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science No. 435.

[CEvdG87] D. Chaum, J.-H. Evertse, and J. van der Graaf. An improved protocol for demonstrating possession of a discrete logarithm and some generalizations. In D. Chaum, editor, *Advances in Cryptology — Eurocrypt '87*, pages 127–141, Berlin, 1987. Springer-Verlag. Lecture Notes in Computer Science No. 304.

[Cha90] D. Chaum. Zero–knowledge undeniable signatures. In I. Damgård, editor, *Advances in Cryptology — Eurocrypt '90*, pages 458–464, Berlin, 1990. Springer-Verlag. Lecture Notes in Computer Science No. 473.

[Cha94] David Chaum. Designated confirmer signatures. In A. De Santis, editor, *Advances in Cryptology — Eurocrypt '94*, pages 86–91, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science No. 950.

[CP92] D. Chaum and T. Pedersen. Wallet databases with observers. In E. Brickell, editor, *Advances in Cryptology — Crypto '92*, pages 89–105, Berlin, 1992. Springer-Verlag. Lecture Notes in Computer Science No. 740.

[CvHP91] D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto '91*, pages 470–484, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science No. 576.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proc. 26th Annual Symp. on the Theory of Computing*, pages 522–533. ACM, 1994.

[DP96] I. Damgard and T. Pedersen. New convertible undeniable signature schemes. In Ueli Maurer, editor, *Advances in Cryptology — Eurocrypt '96*, pages 372–386, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science No. 1070.

[DY91] Y Desmedt and M. Yung. Weaknesses of undeniable signature schemes. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto '91*, pages 205–220, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science No. 576.

[FOO91] A. Fujioka, T. Okamoto, and K. Ohta. Interactive bi-proof systems and undeniable signature schemes. In D. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, pages 243–256, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science No. 547.

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In A. Odlyzko, editor, *Advances in Cryptology — Crypto '86*, pages 186–194, Berlin, 1986. Springer-Verlag. Lecture Notes in Computer Science No. 263.

[GJKR96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In N. Koblitz, editor, *Advances in Cryptology — Crypto '96*, pages 157–172, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science No. 1109.

[GK96] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM. J. Computing*, 25(1), 1996.

[GMR98] R. Gennaro, D. Micciancio, and T Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. To appear in 5th ACM Conference on Computer and Communication Security, 1998.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems . *Journal of the ACM*, 38(1):691–729, 1991.

[Gol95] Oded Goldreich. *Foundation of Cryptography – Fragments of a Book*. Electronic Colloquium on Computational Complexity, February 1995. Available online from *http://www.eccc.uni-trier.de/eccc/*.

[Jak94] M. Jakobsson. Blackmailing using undeniable signatures. In A. De Santis, editor, *Advances in Cryptology — Eurocrypt '94*, pages 425–427, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science No. 950.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In Ueli Maurer, editor, *Advances in Cryptology — Eurocrypt '96*, pages 143–154, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science No. 1070.

[JY96] M. Jakobsson and M. Yung. Proving without knowing: On oblivious, agnostic and blindfolded provers. In N. Koblitz, editor, *Advances in Cryptology — Crypto '96*, pages 201–215, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science No. 1109.

[McC88] K. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology*, 1:95–105, 1988.

[Mic96] M. Michels. Breaking and repairing a convertible undeniable signature scheme. In *ACM Conference on Computer and Communications Security*, 1996.

[Oka94] Tatsuaki Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In Y. Desmedt, editor, *Advances in Cryptology — Crypto '94*, pages 61–74, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science No. 839.

[Ped91] T. Pedersen. Distributed provers with applications to undeniable signatures. In D. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, pages 221–242, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science No. 547.

[Shm85] Z. Shmuely. Composite Diffie-Hellman public key generating systems are hard to break. Technical Report 356, Computer Science Department, Technion, Israel, 1985.