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 self-authenticating) property of digital signatures is not always a desirable property. Such is the case of a signature binding

[*] Extended Abstract. A complete version of the paper is available from `http://www.research.ibm.com/security/papers1997.html`

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

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

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 [CA90] 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 it 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 protection of the recipient is established through the required property. Indeed, 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. In particular, 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 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. 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 signatures. 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, [BCDP91]), *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 that only a specified verifier benefits from interacting with the prover on the confirmation of a signature, [JSI96]). More details 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 [CA90, Cha90, BCDP91, DY91, FOO91, Ped91, CHP92, Cha94, Jak94, Oka94, M96, 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 [BCDP91] 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 [CA90] 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 [BCDP91] and by Damgard and Pedersen [DP96]. In [BCDP91] 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 [BCDP91] was recently broken in [M96]. 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[1] 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.

[1] 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.

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[2]. 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 on the equality of discrete logarithms under different bases is intractable. This assumption is required in previous works as well[3] although by itself 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.

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) 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

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

[3] However, 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.

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 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 [BCDP91]. In doing so we use ideas from the work of Gennaro et al. [GJKR96].

2 Preliminaries

Notation. Throughout the paper we use the following notations:

For a positive integer k we denote $[k] \overset{\text{def}}{=} \{1, \cdots, k\}$. 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 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. The order of elements in Z_n^* is one of the set $\{1, 2, p', q', 2p', 2q', p'q', 2p'q'\}$. Given an element $w \in Z_n^* \setminus \{-1, 1\}$, such that $ord(w) < p'q'$ then $gcd(w - 1, n)$ is a prime factor of n.*

As a consequence of the above lemma we can assume in our protocols that any value found by a party that does not know 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 $m^4 \in <w>$.*

3 Our Undeniable Signature Scheme

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

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

The system is set up by the signer in the following manner: chooses an 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, we state 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)[4]. Thus, the resultant signature of m will be $S_m \overset{\text{def}}{=} \bar{m}^d \bmod n$. In the case of the pair (w, S_w) we will slightly abuse the notation and write $S_w = w^d \bmod n$ (without applying the encoding $\bar{w}$).

[4] 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.

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.

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.

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 $\bar{m}^d$ mod n but $\alpha\bar{m}^d$ mod 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 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 $\mathcal{SIG}(m) \overset{\text{def}}{=} \{S_m : S_m = \alpha\bar{m}^d,\ \ ord(\alpha) \le 2\}$.

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

Fig. 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 [GMW86, 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 $Q \overset{\text{def}}{=} \hat{S}_m^{i} S_w{}^{j} \bmod n$, 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). A proof of the theorem below can be found in [GJKR96].

Theorem 3. Confirmation Theorem. *Let* $(n, w, S_w) \in \mathcal{PK}$.
Completeness. *If P and V follow the Signature Confirmation protocol then V always accepts.*
Soundness. *A cheating prover P^*, even computationally unbounded, cannot convince V to accept $\hat{S}_m \notin \mathcal{SIG}(m)$ with probability greater than $\frac{O(1)}{n}$.*
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.*

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 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 has probability of error $\frac{1}{k}$, where $k = O(\log n)$ is a parameter chosen by the system. Due to an elegant observation of Chaum [Cha90] the desired probability of error can be achieved while incurring only a constant number of exponentiations. He notes that while carrying out k consecutive multiplications, which is equivalent in computation to a single exponentiation, we can compute all the powers in a the range $[k]$. If we take $k = 1024$ 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.
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.

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 $Q_1/Q_2^e = (\frac{m}{\hat{S}_m^e})^i$ and computes i by testing all possible values of $i \in [k]$.
   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.

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

Theorem 4. 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* $\frac{1}{k} + \frac{1}{n}$*.*

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.*

4 Security Analysis

We do not present here a formal treatment of the notion of undeniable signatures and its security requirements. 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 in an informal way based on this outline.

4.1 Unforgeability of signatures

We consider an attacker that cannot forge regular RSA signatures. When attacking our undeniable signatures scheme this attacker may request signatures (and their confirmation) on any messages of its choice. The attacker can also choose pairs of messages and alleged signatures and engage in confirmation or denial protocols with the signer on these inputs (whether it engages in a confirmation or denial protocol depends on the validity or invalidity, respectively, of the input pair). The goal of the attacker is to *forge* a signature, namely, to generate a valid signature on a message not previously signed by the legitimate signer.

We first note that since both confirmation and denial protocols are zero-knowledge then the information provided to the attacker by these protocols is useless for attacking the signatures (in the sense that the same information can be generated by the attacker alone). Therefore, an attacker could essentially try to forge signatures based on the public keys and a (possibly chosen) list of messages and their valid signatures. However, since our signatures are equivalent to regular RSA signatures (except for the fact that the verification exponent is secret which can only make it harder for the attacker) then the ability to forge our undeniable signatures would translate into forging regular RSA signatures which we assume infeasible. (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.)

Formalizing the above arguments is quite straightforward and standard. Such a formal proof would show how to transform any given forging attacker against our undeniable signatures into a forging attacker against regular RSA signatures; the transformation would make use of the simulators for our zero-knowledge protocols (both confirmation and denial). We summarize this discussion in the following theorem.

Theorem 5. *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).*

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

$$\log_{\bar{m}}(s(m)) \overset{?}{=} \log_w(S_w) \tag{1}$$

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. 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 1 over Z_n^*.

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 in the case of [DP96] the EDL assumption is not sufficient to prove simulatability, which in that paper is indeed simply conjectured.

Theorem 6. *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 more common signing/verification operations. We 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 [CEG87], 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. We have three alternative solutions for this problem. The first is to use a generic zero-knowledge proof of the above property using the general results of [GMW86]; although the resultant solution would be highly inefficient this task is performed only once at system initialization. A more efficient (but less secure) solution to this problem is to let the signer generate a large set of moduli $n_1, n_2, \cdots, n_k$ from which V chooses a random element, say n_i. Next, P shows the factorization into primes of all the other moduli in the set. If all are of the right form then n_i is chosen as the modulus n, otherwise P

is disqualified. The drawback of this solution is that the probability of cheating, i.e. $1/k$, reduces only linearly with the amount of work in the protocol. Yet once again, the protocol needs to be performed only at initialization of the modulus and thus a relatively large number k of moduli can be produced. (Although this gives only "linear security" we stress that under the appropriate legal circumstances a probability of, say 999/1000, to be caught cheating can be a significant deterrent for anyone to register an invalid key.)
Finally, there is a solution [Dam] that allows for a trade-off between the error probability at the key registration stage and the performance cost of the undeniable signature scheme. Initially, we let P generate $2k$ moduli. V chooses at random k of them of which the signer must reveal the factorizations. If the factorization of those moduli was of the correct form, we run our basic scheme in parallel for all the remaining k moduli. A confirmation or denial is accepted only if it works for all k moduli. The signer can only cheat if all the opened moduli were good, and all the remaining bad, but for any given set of moduli, this will only happen with exponentially small probability in k. In practice, one can choose parameters more appropriately. For example it doesn't have to be $2k$ and k moduli since with a total of 100 moduli, V choosing 95 of them and keeping only 5 to do the scheme in parallel, the error probability is close to 10^{-9}.

5 Extensions

Our protocols lend themselves to many of the existing extensions in the literature for undeniable signatures.
Convertible Undeniable Signatures. This variation appeared first in [BCDP91], 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.[5]

[5] 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 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.

In some applications it may be desirable to convert only a subset of the past signatures (*selective conversion* [BCDP91]). For this scenario we can make use of a non-interactive zero-knowledge confirmation proof for those messages. Such an efficient scheme is described in the final paper.

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 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.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.

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.

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 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).
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.

Acknowledgments. We would like to thank 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.

[BCDP91] J. Boyar, D. Chaum, I. Damgård, and T. Pedersen. Convertible undeniable signatures. In A.J. Menezes and S. A. Vanstone, editors, *Proc. CRYPTO 90*, pages 189–205. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 537.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures, how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology: EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[CA90] David Chaum and Hans Van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 212–217. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[CEG87] D. Chaum, J.-H. Evertse, and J. van der Graaf. An improved protocol for demonstrating possession of a discrete logarithm and some generalizations. In *EUROCRYPT'87*, pages 127–141, 1987.

[Cha90] D. Chaum. Zero–knowledge undeniable signatures. In *Proc. EUROCRYPT 90*, pages 458–464. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 473.

[Cha94] David Chaum. Designated confirmer signatures. In *EUROCRYPT'94*, pages 86–91, 1994.

[CP93] D. Chaum and T. Pedersen. Wallet databases with observers. In *CRYPTO'92*, pages 89–105. Springer-Verlag, 1993. Lecture Notes in Computer Science No. 740.

[CHP92] D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In J. Feigenbaum, editor, *Proc. CRYPTO 91*, pages 470–484. Springer, 1992. Lecture Notes in Computer Science No. 576.

[Dam] I. Damgård. Personal communication. November, 1996.

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

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

[DY91] Y Desmedt and M. Yung. Weaknesses of undeniable signature schemes. In *Eurocrypt'91*, pages 205–220, 1991.

[FOO91] A. Fujioka, T. Okamoto, and K. Ohta. Interactive bi-proof systems and undeniable signature schemes. In *Eurocrypt'91*, pages 243–256, 1991.

[FS86] Fiat, A. and Shamir, A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems.. In *Crypto'86*, pages 186–194. Springer-Verlag, 1986. 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 *Crypto'96*, pages 157–172. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1109. Complete version available from `http://www.research.ibm.com/security/papers1997.html`

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but the Validity of the Assertion, and a Methodology of Cryptographic Protocol Design. In *Proceeding 27th Annual Symposium on the Foundations of Computer Science*, pages 174–187. ACM, 1986.

[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 *EUROCRYPT'94*, pages 425–427, 1994.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. Maurer, editor, *Advances in Cryptology: EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 1996.

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

[M96] M. Michels. Breaking and Repairing a Convertible Undeniable Signature Scheme. In Proceedings of the 1996 ACM Conference on Computer and Communications Security, 1996.

[Oka94] Tatsuaki Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In Yvo G. Desmedt, editor, *Advances in Cryptology: CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 61–74. Springer-Verlag, 1994.

[Ped91] T. Pedersen. Distributed provers with applications to undeniable signatures. In *Eurocrypt'91*, pages 221–242, 1991.