RSA-based Undeniable Signatures For General Moduli

Steven D. Galbraith[1], Wenbo Mao[2] and Kenneth G. Paterson[3]

[1] Pure Mathematics Department, Royal Holloway University of London,
Egham, Surrey TW20 0EX, UK.
Steven.Galbraith@rhul.ac.uk

[2] Mathematics, Cryptography and Security Group
Hewlett-Packard Laboratories, Bristol
Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK.
wm@hplb.hpl.hp.com

[3] Information Security Group,
Mathematics Department, Royal Holloway University of London,
Egham, Surrey TW20 0EX, UK.
Kenny.Paterson@rhul.ac.uk

Abstract. Gennaro, Krawczyk and Rabin gave the first undeniable signature scheme based on RSA signatures. However, their solution required the use of RSA moduli which are a product of safe primes. This paper gives techniques which allow RSA-based undeniable signatures for general moduli.

Keywords: Undeniable Signatures, RSA-based Undeniable Signatures.

1 Introduction

Undeniable signatures were introduced by Chaum and van Antwerpen [7, 8]. They offer good privacy for the signer since signatures cannot be verified without interaction with the signer. Undeniable signatures (and generalisations of them, such as confirmer and convertible signatures) have various applications in cryptography [2, 3, 10].

The zero-knowledge undeniable signature scheme of Chaum [8] works in the multiplicative group of integers modulo a prime. Although Chaum, van Heijst and Pfitzmann [9] provided an undeniable signature scheme with security related to factoring, before 1997 there was not a scheme based on traditional RSA signatures. Gennaro, Krawczyk and Rabin [18] were the first to obtain an RSA-based undeniable signature scheme. Their scheme is closely related to the scheme of Chaum [8] and both schemes have similar security and efficiency.

One significant drawback with the scheme of [18] is that it requires the use of RSA moduli which are products of safe primes. It was explicitly stated as an open problem in [18] to provide an undeniable signature scheme based on RSA which does not require special moduli. The goal of the present paper is to solve this problem.

Of course, it is trivial to construct an undeniable signature scheme for general moduli where the confirmation protocol has soundness probability $1/2$, but we seek solutions where the confirmation protocol is more efficient (possibly at the expense of more demanding key certification). We must mention that general constructions due to Michels and Stadler [23] and Camenisch and Michels [5] also give solutions to this problem, however their systems require auxiliary tools ([23] utilises confirmer commitment schemes, while [5] requires a secure encryption scheme).

In the course of solving this problem we improve the efficiency and zero-knowledge property of the denial protocols for RSA-based undeniable signatures. The methods of this paper are therefore a useful addition to the protocol of [18], even when safe primes are being used.

1.1 Pros and cons of special moduli

The undeniable signature scheme of Gennaro, Krawczyk and Rabin [18] is modelled on RSA [27]. Thus a signature on a message m is a number $s = m^d \pmod{N}$ where N is a product of two primes and where d is an integer coprime to $\varphi(N)$. The difference between usual RSA signatures is that the number e such that $de \equiv 1 \pmod{\varphi(N)}$ is not public, and so an interactive proof (preferably zero-knowledge) is required to confirm that s is a valid signature for m (i.e. that $s^e \equiv m \pmod{N}$). This protocol relies on a public key having previously been certified by an authority.

There are various reasons why Gennaro, Krawczyk and Rabin [18] restricted to the case where the RSA modulus N is a product of safe primes (i.e. primes p such that $(p-1)/2$ is also prime) but the most important one is that, for products of safe primes, the group $\mathbb{Z}_N^*$ does not have many elements of small order. If one runs the scheme of Gennaro, Krawczyk and Rabin [18] with a general modulus then there is a high probability that a dishonest signer can cheat (see Section 3 for details).

In general, restricting to moduli which are a product of safe primes makes many cryptographic issues easier to handle. However, there are several drawbacks of schemes which require special moduli. One major problem is that it is necessary for a certification authority to guarantee the properties of the public key. As we discuss next, none of the currently known protocols for allowing a user to prove to a certification authority that their modulus is a product of safe primes are fully satisfactory.

Gennaro, Micciancio and Rabin [17] have given a very nice protocol to prove the a number is a product of two quasi-safe primes (i.e. primes p such that $(p-1)/2$ is a power of a prime), but this is significantly less than the assurance we require. For instance, a prime of the form $2 \cdot 3^k + 1$ is a quasi-safe prime but a modulus constructed as a product of primes of this form would be vulnerable to attacks such as those outlined in Section 3.

Camenisch and Michels [4] have given a protocol to prove that a number is a product of safe primes. Their protocol requires performing the Miller-Rabin primality test in zero-knowledge on a hidden number. It therefore requires an enormous amount of communication between the prover and the certification authority. This protocol is unsuitable for practical applications.

Another problem is that choosing special moduli goes against the conventional wisdom in cryptography of avoiding special cases. Indeed, in Section 8.2.3 of [22] and in [29] it is explicitly stated that products of random primes are advisable for cryptography.

There are many other protocols which currently require moduli which are a product of safe primes [6, 14, 16, 26, 28] and it is of great interest in cryptography to provide solutions which do not require this assumption. Some recent papers in this direction include [13] and [15]. We hope that the new techniques introduced in this paper might be of wider applicability to solve other problems in the area.

1.2 Our Work

We provide an undeniable signature scheme for general RSA moduli. Our scheme is, in fact, a parameterised family of cryptosystems depending on three parameters B, K_1 and K_2.

The number 2^{-K_1} will be the probability that a dishonest signer Alice will be able to cheat the certification authority (CA) when certifying her public key. A value of K_1 should be agreed in advance between Alice and the CA and could form part of Alice's certified public key. Similarly 2^{-K_2} will be the probability that Alice can cheat a verifier Bob in either the confirmation or denial protocols. We allow different values of K_1 and K_2 for generality. Typical values that might be used in practice are $K_1 = K_2 = 100$.

The number $1/B$ will be the soundness probability for each iteration of the signature confirmation and denial protocols. The number of iterations required to obtain a cheating probability of 2^{-K_2} will be $\frac{K_2}{\log_2 B}$. A typical choice of B might be 2^{10} in which case 10 iterations of our protocols are needed for $K_2 = 100$.

The value of B also determines how 'special' the moduli must be, and accordingly, how expensive public key certification is. Essentially, with B chosen, the modulus N must have the property that $\varphi(N)$ is not divisible by any odd primes $p < B$. Alice will prove this to the CA during key certification.

Large values of B will give efficient signature and denial protocols, but the moduli N will be rather special (and there is necessarily a lot of work required in our process for public key certification). In some sense, moduli which are a product of safe primes as in [18] are a limiting case of our cryptosystem in which $B = N^{1/4}$. Our public key certification process has been designed with rather general RSA moduli in mind (i.e. for small values of B). If special RSA moduli (i.e. larger values of B) are to be used then certification protocols should be developed using techniques like those in [5]. Small values of B result in a scheme which does not require special moduli (and for which public key certification is relatively efficient), but the resulting confirmation and denial protocols require many rounds to achieve the desired soundness probability of 2^{-K_2}. We therefore have a tunable family of undeniable signature schemes. In particular, we do obtain an undeniable signature scheme which works for completely general

RSA moduli (see Section 8.4). For a fuller discussion of the performance of our schemes, see Section 8.3. As we shall see, for the typical values $K_1 = K_2 = 100$ and $B = 2^{10}$, the protocols are all perfectly practical.

The next section sets up some notation for the rest of the paper. In Section 3 we review the scheme of [18] and indicate some of the pitfalls in adapting this scheme to general RSA moduli. Our process for public key certification is specified in detail in Section 4. We emphasise that the cost of certification is a one-time cost. Our signature confirmation and denial protocols are described in Section 5, with proofs of zero-knowledge and security against existential forgery appearing in the following two sections. One important innovation here is a new signature denial protocol which is more efficient and has a cleaner proof of zero-knowledge than the protocol used in [18]. In Section 8 we give variations of the scheme which provide confirmer signatures and convertible signatures. We also discuss the performance of our scheme there.

2 Preliminary definitions and notation

Let N be a positive integer. We write $\mathbb{Z}_N^*$ for the multiplicative group of integers modulo N. We write Q_N for the subgroup of quadratic residues (squares) in $\mathbb{Z}_N^*$. We write $\varphi(N)$ for the Euler phi function. A safe prime is an odd prime p such that $(p-1)/2$ is prime.

Given any $g \in \mathbb{Z}_N^*$ we define the order of g to be $\mathrm{ord}(g) = \min\{n \in \mathbb{Z} : n \geq 1 \text{ and } g^n \equiv 1 \pmod{N}\}$. When $N = p_1 p_2$ is a product of two distinct primes then every $g \in \mathbb{Z}_N^*$ has order dividing the least common multiple $\mathrm{lcm}(p_1 - 1, p_2 - 1)$.

3 The scheme of Gennaro, Krawczyk and Rabin

In this section we briefly sketch the RSA-based undeniable signature scheme of Gennaro, Krawczyk and Rabin [18] for products of safe primes. We also indicate why it is nontrivial to adapt this to the case of a general RSA modulus.

Alice possesses a public RSA modulus N, which is assumed to be a product of two safe primes, and a pair of secret integers (e, d) such that $ed \equiv 1 \pmod{\varphi(N)}$. Alice's undeniable signature on a message $m \in \mathbb{Z}_N$ is $s = m^d \pmod{N}$, i.e. a standard RSA signature.

Since e is not public knowledge, it is not possible for Bob to verify the validity of the signature s without interacting with Alice. Instead, Alice the prover and Bob the verifier engage in a zero-knowledge protocol to show that $s^e \equiv m \pmod{N}$. For this signature confirmation protocol it is necessary to have some fixed commitment to the value e. This is achieved in [18] by taking a random element $g \in \mathbb{Z}_N^*$ (which can be shown to have large order in the case of special moduli) and publishing $h = g^d \pmod{N}$

The signature confirmation protocol of [18] (presented for simplicity in the case of honest verifiers) is the following:

1. Given the public key (N, g, h) and an alleged message-signature pair (m, s) the verifier chooses random integers $1 \leq i, j < N$, constructs a challenge $C = s^{2i}h^j \pmod N$, and sends C to the prover.
2. The prover sends the response $R = C^e \pmod N$ to the verifier.
3. The verifier checks whether $R \equiv m^{2i}g^j \pmod N$.

The signature denial protocol suggested in [18] is an adaptation of a protocol due to Chaum (originally developed for the case of finite fields $\mathbb{F}_q^*$). The denial protocol requires the prover to perform an exhaustive search over k values where k is a security parameter. The probability of successful cheating by a dishonest prover in this case is $1/k$. There is also a minor complication about how aborting the protocol affects the zero-knowledge properties, this is handled in [18] by using a commitment to zero.

3.1 Generalising to general moduli, problem I

In this and the next subsection we motivate the need for our more complex protocols by considering what happens if the protocol due to [18] is naively used with a general RSA modulus N. The problems we sketch should be seen as part of a general phenomenon, that protocols developed in the case of finite fields do not necessarily give rise to secure protocols when working with $\mathbb{Z}_N^*$.

Let Alice be a dishonest prover. Since Alice controls the factorisation of the modulus N she can choose N so that there is a small prime ℓ with $\ell | \varphi(N)$. She can also find an element $\alpha \in \mathbb{Z}_N^*$ such that α has order ℓ. Suppose Alice publishes a signature $s = \alpha m^d$ for a message m. What is the probability that Alice can fool a verifier Bob that this is a valid signature? In the confirmation protocol Alice receives a challenge $C = s^{2i}h^j \pmod N$. In general Alice does not know the value of i, but she can compute a response $R = \alpha^r C^e \pmod N$ where r is chosen at random. If $r + 2ie \equiv 0 \pmod \ell$ then the check performed by the verifier will be satisfied. Hence the probability of successful cheating is at least $1/\ell$. Since ℓ can be chosen to be 3 this probability is quite high. There is an analogous attack using elements of order 4 which has probability 1/2 of success. Hence the confirmation protocol must be executed many times to give an assurance that the signature is valid. This is unsatisfactory.

Notice that when N is a quasi-safe prime product (see [17]), using a small ℓ as above will render N vulnerable to well-known factoring algorithms, such as Pollard's $P - 1$ method or the elliptic-curve method. So if Alice's objective is to fool Bob with reasonable probability and she is not concerned about using a modulus that succumbs to these factoring algorithms, then she can choose to use a modulus that is a product of quasi-safe primes.

We will solve these issues by giving a method for Alice to certify that her public key N is such that there are no small (up to a bound B) odd primes dividing $\varphi(N)$.

3.2 Generalising to general moduli, problem II

There is a more subtle and devastating attack. Once again suppose Alice is a dishonest signer and suppose that (either by construction, or by accident) her public key element g does not have maximal order in $\mathbb{Z}_N^*$.

For simplicity of presentation we suppose that there is a prime q such that $q \| \varphi(N)$ (i.e. $q | \varphi(N)$ but $q^2 \not| \varphi(N)$) and $q \not| \operatorname{ord}(g)$. We assume that q is not too large (less than 80 bits, say) so that the discrete logarithm problem in the subgroup of order q can be solved using standard methods. Let $\alpha \in \mathbb{Z}_N^*$ be an element of order q. Alice constructs her public key $h = g^d \pmod{N}$ as usual.

Let $m \in \mathbb{Z}_N^*$ be any message (it doesn't matter whether $q | \operatorname{ord}(m)$ or not). Suppose Alice publishes $s = \alpha m^d \pmod{N}$ as her signature on m.

Consider the signature confirmation protocol. Alice receives the challenge $C = s^{2i} h^j$. By raising C to the power $\varphi(N)/q$ and solving a discrete logarithm problem to the base α Alice can determine the value of $i \pmod{q}$. Alice can therefore respond with $R = \alpha^r C^e \pmod{N}$ where $r = -2ie \pmod{q}$ is constructed so that the check by the verifier will always be satisfied. In other words, Alice can fool the verifier with probability one! Similarly, whenever Alice desires, she can successfully run a signature denial protocol on that signature.

This is an extremely severe attack on an undeniable signature scheme. We address this problem in our scheme by using a set of generators $g_1, \ldots, g_k$ where we take k to be large enough so that the group generated by all the g_i is overwhelmingly likely to contain Q_N.

4 Public key certification

Suppose Alice wants to be able to generate undeniable signatures. Let the parameters B and K_1 be fixed as in Section 2. The public key for Alice is a tuple $(N, g_1, \ldots, g_k, h_1, \ldots, h_k)$ where k is such that $g_1, \ldots, g_k$ generate a subgroup of $\mathbb{Z}_N^*$ which contains Q_N with probability at least $1 - 2^{-K_1}$. For example, for the typical values $B = 2^{10}$ and $K_1 = 100$ we can take $k = 11$. More generally, we should take k so that $\frac{2}{k-1}(B-1)^{1-k} < 2^{-K_1}$ (see below). The private key is a pair (e, d) (these values are also defined below). We emphasise that this is different from a standard RSA public key, which would include the signature verification exponent e. Alice must register her public key with a CA, who will issue a certificate which confirms that Alice's public key is suitable for the undeniable signature scheme we propose. The properties of the public key which must be guaranteed by this certificate are:

1. N is a product of two prime powers $p_1^{s_1} p_2^{s_2}$ such that each $p_i \equiv 3 \pmod{4}$. (See Section 8.4 for discussion of how to relax the assumption that $p_i \equiv 3 \pmod{4}$.)
2. $\gcd(\Delta, \varphi(N)) = 1$ where Δ is the product of all primes $2 < l < B$.
3. The g_i are chosen at random in a way which is not controlled by Alice.
4. The g_i and h_i are correctly related by $h_i = g_i^d \pmod{N}$ for some secret integer d which is coprime to $\operatorname{lcm}(\operatorname{ord}(g_1), \ldots, \operatorname{ord}(g_k))$.

4.1 Construction of the modulus

The first step of key generation for Alice is to construct an integer $N = p_1 p_2$ which is a product of two primes such that $p_i \equiv 3 \pmod 4$. Let B be the integer specified in Section 1.2 and which determines the soundness probability of our confirmation and denial protocol. We demand that for all primes $2 < l < B$ one has $l \not| (p_i - 1)$ for $i \in \{1, 2\}$. This means that $\varphi(N)$ is coprime to $\Delta = \prod_{\text{primes } 2<l<B} l$.

Alice must prove to the CA that N is a product of primes p_i such that $p_i \equiv 3 \pmod 4$. This can be achieved using a protocol due to van de Graaf and Peralta [19]. The protocol of [19] proves that $N = p_1^{s_1} p_2^{s_2}$ where $p_i \equiv 3 \pmod 4$ and the s_i are odd integers. This is enough for our application (we do not need to assume that $s_1 = s_2 = 1$ for our protocols) and if a stronger assurance about the difficulty of factoring N is required then more advanced techniques may be used.

4.2 Proof that $\varphi(N)$ does not have small prime factors

Alice must prove to the CA that $\varphi(N)$ is not divisible by any odd primes $2 < l < B$. In Figure 1 we give a protocol to achieve this in the honest verifier case. Recall that Δ is the product of all primes $2 < l < B$. So Δ is approximately B bits in length (for $B = 2^{10}$, Δ has 1420 bits). The protocol involves exponentiations to the power Δ. For small B, say up to 2^{10}, this is a perfectly feasible computation. However, for large B (necessary when a modulus is being certified to be 'special'), this becomes infeasible. The largest B one might wish to use in practice would be, perhaps, $B = 2^{20}$.

Protocol Certification–1(N, Δ).

1. The CA chooses a random integer $x \in \mathbb{Z}_N^*$, computes the challenge $C = x^{\Delta} \pmod N$ and sends C to Alice.
2. Alice sends to the CA the response $R = C^{\Delta^{-1}} \pmod N$ which is unique when $\gcd(\varphi(N), \Delta) = 1$.
3. The CA accepts if $R = x$.

Fig. 1. Proof that $\gcd(\Delta, \varphi(N)) = 1$ in the honest verifier case.

This protocol can be made into a perfect zero-knowledge protocol (i.e. robust against dishonest verifiers) in a standard way: Replace the second move (i.e. where Alice sends her response to the CA) with the transmission of a perfectly-hiding commitment to R. The CA then must send x to Alice, so that she can check that the initial challenge was correctly formed. Alice can then open the commitment to R, allowing the CA to see that the response is correct.

Theorem 1. *Let (N, Δ) be as above. The protocol Certification–1 has the following properties:*

Completeness: *If* $\gcd(\Delta, \varphi(N)) = 1$ *then the CA will always accept Alice's proof.*

Soundness: *If* $\gcd(\Delta, \varphi(N)) \neq 1$ *then Alice, even computationally unbounded, cannot convince the CA to accept her proof with probability better than* $1/3$.

Zero-knowledge: *When Alice behaves correctly, the CA gains no information about Alice's private input apart from the validity of her proof.*

Proof. The proof of completeness is immediate from inspection of the protocol. We focus on the further two properties.

Soundness: Suppose that l is a prime such that $l \mid \gcd(\Delta, \varphi(N))$. Then there are at least l elements of $\mathbb{Z}_N^*$ of order l (there could be l^2 of them). Let $\alpha \in \mathbb{Z}_N^*$ be such an element. Let $x \in \mathbb{Z}_N^*$ be chosen at random and define $C = x^\Delta \pmod{N}$. Then for each $0 \leq i < l$ we have $C \equiv (x\alpha^i)^\Delta \pmod{N}$. Alice therefore has no information about which of the possibilities x is the one chosen by the CA. Hence Alice cannot respond with the correct value with probability better than $1/l$.

This discussion applies to all primes $2 < l < B$, but in the worst case (as far as the CA knows) we have $\gcd(\Delta, \varphi(N)) = 3$.

Zero-knowledge: Alice publishes a perfectly hiding commitment to the value R and only opens it if the CA already knows the value. Hence the CA learns nothing. □

The protocol must be repeated $\frac{K_1}{\log_2 3}$ times to achieve a probability of successful cheating as low as 2^{-K_1}. The flows of instances of the protocol may all be sent in parallel.

It would be very interesting to have an efficient protocol to prove that $\varphi(N)$ is coprime to the integer Δ which has a soundness probability smaller than $1/3$. It seems to be highly non-trivial to construct such a protocol.

4.3 Construction of generators

The next step of key generation is to construct elements $g_1, \ldots, g_k \in \mathbb{Z}_N^*$. We cannot allow Alice to generate these elements as she might choose them to have small order, in which case attacks like those in Section 3 apply. Hence the values for the g_i should be generated using a protocol in which both Alice and the CA jointly contribute randomness (this is easy to achieve using commitment schemes). Another solution would be to let the CA choose the values g_i (indeed, they could even be fixed values for all users).

The reason for choosing many elements g_i is to ensure that the whole of the subgroup $Q_N \subset \mathbb{Z}_N^*$ is generated (see the attack in Section 3.2). A similar technique has been used in [25], [15]. This will be important in the soundness proof of our signature confirmation protocol.

With $N = p_1 p_2$ the techniques in [25] and [15] can be used to show that the number of k-tuples $g_1, \ldots, g_k$ generating all of $Q_N \subset \mathbb{Z}_N^*$ is equal to

$$\varphi_k((p_1 - 1)/2)\varphi((p_2 - 1)/2)$$

where

$$\varphi_k\left(\prod_{i=1}^{t} q_i^{e_i}\right) = \prod_{i=1}^{t} q_i^{e_i} \prod_{i=1}^{t} (1 - q_i^{-k})$$

is a generalisation of the Euler phi function. From this, using estimates like those in [25], we can prove that the probability that $g_1, \ldots, g_k$ generate all of Q_N is at least

$$1 - \frac{2}{k-1}(B-1)^{1-k}$$

where B is the bound on the size of the prime factors of $(p_1 - 1)/2$ and $(p_2 - 1)/2$. We can make this probability arbitrarily close to one by taking k to be sufficiently large. For example, with $B = 2^{10}$ we can take $k = 11$ to guarantee that the g_i generate $Q_N \subset \mathbb{Z}_N^*$ with probability at least $1 - 2^{-100}$

In any case, we assume from now on that k has been chosen large enough that the g_i generate a subgroup of $\mathbb{Z}_N^*$ which contains Q_N.

The next step is for Alice to choose a secret key pair (e, d) such that $ed \equiv 1 \pmod{\varphi(N)}$. Alice should then construct $h_i = g_i^d \pmod{N}$ for $i = 1, \ldots, k$.

It is crucial that the h_i are correctly calculated and so Alice must provide a proof that this is the case. We give such a proof in Figure 2.

Protocol Certification–2$(N, g_1, \ldots, g_k, h_1, \ldots, h_k)$.

1. Alice chooses random integers $1 \leq k_1, k_2 \leq \varphi(N)$, constructs $u_i = h_i^{e+k_1} \pmod{N}, v_i = g_i^{d+k_2} \pmod{N}$ for $i = 1, 2, \ldots, k$, and sends the elements u_i and v_i to the CA.
2. The CA sends to Alice a random bit $b \in \{0, 1\}$.
3. If the bit is zero then Alice sends the two integers $n_1 = (e + k_1) \pmod{\varphi(N)}$ and $n_2 = (d + k_2) \pmod{\varphi(N)}$. The CA can then check whether $u_i \equiv g_i^{n_1} \pmod{N}$ and $v_i \equiv g_i^{n_2} \pmod{N}$ for all $i = 1, 2, \ldots, k$. The CA accepts if all the checks pass and rejects otherwise.
   If the bit is one then Alice sends the two integers k_1 and k_2. The CA then checks whether $u_i \equiv g_i h_i^{k_1} \pmod{N}$ and $v_i \equiv h_i g_i^{k_2} \pmod{N}$ for $i = 1, 2, \ldots, k$. The CA accepts if all the checks pass and rejects otherwise.

Fig. 2. Key certification protocol for g_i, h_i.

Theorem 2. *Let (N, g_i, h_i) be as above. For $N = p_1^{s_1} p_2^{s_2}$ define M to be the exponent of the group $\mathbb{Z}_N^*$ (i.e., $M = \lambda(N)$). Assume that the group generated by the g_i contains Q_N (and so, in particular, $(M/2) | \mathrm{lcm}(\mathrm{ord}(g_1), \ldots, \mathrm{ord}(g_k))$). The protocol Certification–2 has the following properties:*

Completeness: *If* $h_i \equiv g_i^d \pmod{N}$ *where* $ed \equiv 1 \pmod{M/2}$ *then the CA will accept Alice's proof.*

Soundness: *If some* $h_i \not\equiv g_i^d \pmod{N}$ *or if* $ed \not\equiv 1 \pmod{M/2}$ *then Alice, even computationally unbounded, cannot convince the CA to accept her proof with probability better than* $1/2$*.*

Zero-knowledge: *When Alice behaves correctly, the CA gains no information about Alice's private input apart from the validity of her proof.*

Proof. The proof of completeness is immediate from inspection of the protocol. We focus on the further two properties.

Soundness: If Alice can respond correctly to both choices of the bit b then she knows numbers e and d such that $h_i \equiv g_i^d \pmod{N}$ and $g_i \equiv h_i^e \pmod{N}$ for all $i = 1, 2, \ldots, k$. This proves that all the g_i and h_i are related by the same numbers.

We now show that the protocol implies $ed \equiv 1 \pmod{M}$. Suppose that there is some odd prime power $q^a | M$ such that $ed \equiv x \not\equiv 1 \pmod{q^a}$. By assumption, $q^a | \mathrm{ord}(g_i)$ for some index i. Let $g = g_i^{(\mathrm{ord}(g_i)/q^a)}$ and $h = h_i^{(\mathrm{ord}(g_i)/q^a)}$. We have $g^x \equiv g^{ed} \equiv h^e \equiv g \pmod{N}$ which is a contradiction.

Zero-knowledge: This is immediate, a standard argument showing that transcripts of the protocol can be simulated. □

The protocol must be repeated K_1 times to achieve a probability of successful cheating to be 2^{-K_1}. The flows of instances of the protocol may all be sent in parallel.

It would be very useful to have a more efficient protocol to prove the correctness of the g_i and h_i. It seems to be non-trivial to find such a protocol. The standard methods used when working in finite fields $\mathbb{F}_q^*$ do not immediately translate into secure protocols to solve our problem – they are vulnerable to attacks of the type described in Section 3.

5 Undeniable RSA Signatures

As usual with RSA it is not possible to allow any number $m \in \mathbb{Z}_N^*$ to be a valid message. Instead we need to use a cryptographically strong randomised padding scheme to provide an integrity check on messages. We return to this issue in Section 7, but for now we simply assume that this has been performed and that we want to provide a signature for some element $m \in \mathbb{Z}_N^*$.

Alice's signature on m is the usual RSA signature $s = m^d \pmod{N}$. There is one technicality as the signature confirmation protocol does not distinguish between signatures which differ by an element of order dividing 2. This is also the situation in [18]. The problem is easily solved allowing all four values s such that $s^{2e} \equiv m^2 \pmod{N}$ to be valid signatures.

5.1 Confirmation of an undeniable signature

Let (m, s) be an alleged signature pair. In Figure 3 we give a protocol (in the honest verifier case) for Alice to prove to a verifier Bob that the alleged pair is a genuine one. We assume that Alice has sent (m, s) and her certified public key information to Bob. There are two solutions to make the protocol robust against dishonest verifiers (i.e. perfect zero-knowledge) and we discuss them shortly.

Protocol Confirm$(N, g_1, \ldots, g_k, h_1, \ldots, h_k, m, s)$.

1. Bob chooses random integers $r_0, r_1, \ldots, r_k$ such that all $1 < r_i < N$. Bob computes the challenge $C = s^{r_0} h_1^{r_1} \cdots h_k^{r_k} \pmod{N}$.
2. Bob sends to Alice the challenge C.
3. Alice sends to Bob the response $R = C^e \pmod{N}$.
4. Bob accepts if $R^2 \equiv (m^{r_0} g_1^{r_1} \cdots g_k^{r_k})^2 \pmod{N}$ and rejects otherwise.

Fig. 3. Undeniable signature confirmation protocol in the honest verifier case

The security of this protocol is discussed in Section 6. The protocol must be repeated $K_2/\log_2 B$ times (the flows of each instance may be executed in parallel) to achieve the desired probability of $1 - 2^{-K_2}$ that the signature is valid.

We now discuss how to transform this protocol into a perfect zero-knowledge protocol (i.e. robust against dishonest verifiers). The first solution uses the standard technique: Instead of sending the response R, Alice publishes a commitment to it, and opens the commitment only once Bob has shown that the challenge C is of the correct form. We emphasise that the zero-knowledge property of the protocol only holds when the input is a valid message-signature pair (otherwise the protocol gives a message corresponding to a given signature). A signature confirmer must be careful to only execute the confirmation protocol on valid inputs (the denial protocol given in the next section should be used in other cases).

We note that, for undeniable signatures, it is usually preferable to use designated verifier proofs in protocols such as the one above. This can be easily achieved using the methods of Jakobsson, Sako and Impagliazzo [20].

There is another solution which provides security only against forgery of signatures. Instead of sending the response R, Alice sends $t = H(R^2 \pmod{N})$ where H is some cryptographically strong hash function. Bob can then check whether or not t is equal to $H((m^{r_0} g_1^{r_1} \ldots g_k^{r_k})^2 \pmod{N})$. This proof method does not allow Bob to use Alice as a signing oracle, which prevents Bob from being able to forge signatures. The main problem with this method is that Alice no longer knows which message signature pair (m, s) she is being requested to verify. This is an attack on an undeniable signature scheme since one intended feature of these schemes is that Alice should be aware which of her signatures are being verified. Nevertheless, in some contexts this proof technique might be of use.

5.2 Denial of an undeniable signature

Given a pair (m, s) which is not a valid signature for Alice on the message m (i.e. $s \not\equiv \xi m^d \pmod{N}$ where $\mathrm{ord}(\xi)|2$), it is important to provide a protocol allowing Alice to prove that this is the case.

One way to achieve this would be to compute the true signature $m^d \pmod{N}$ for m, send it to the verifier, and then prove its correctness using the signature confirmation protocol above. But this leaves the user open to a chosen-message attack, since she can be forced to publish valid signatures on any message m.

Instead, we proceed by allowing Alice to modify the alleged pair (m, s) to obtain a random related pair (m', s') and then perform the above process. To the eye of a seasoned cryptographer this still looks like a security flaw, but it is well-known that most undeniable signature schemes (e.g. [7], [18]) allow existential forgery using just the public key. These issues are handled using padding schemes on messages (further discussion is given in Section 7).

Our denial protocol runs as follows: Bob (or Alice and Bob running a joint protocol) chooses a random integer $1 < r < N$. Both parties can then compute the related elements $m' = m^r \pmod{N}$ and $s' = s^r \pmod{N}$. Alice publishes her correct signature s'' for the message m' and proves it is correct using the confirmation protocol above. The verifier can then determine whether $(s')^2 \equiv (s'')^2 \pmod{N}$.

The denial process requires only two more exponentiations than the confirmation protocol per participant, and the security is the same as that of the confirmation protocol. This is in contrast with Gennaro, Krawczyk and Rabin [18] where the denial protocol is much less efficient than the confirmation protocol. There is no loss of security from performing signature denial using our methods and so the protocol of [18] can be improved by using our approach. We note that Miyazaki [24] also gave an efficient denial protocol for this application based on the denial protocol of Chaum and van Antwerpen [7], although that protocol requires double the computation of ours.

Note that this denial method cannot be used in the classical case of undeniable signatures in a finite field $\mathbb{F}_q^*$ since the verifier can undo the transformation of exponentiation by r to recover a signature on a chosen message. We note that Jakobsson [21] has given an efficient denial protocol for Chaum's undeniable signatures. It is straightforward to adapt Jakobsson's ideas to the case of RSA-based undeniable signatures, but we obtain a denial protocol which takes at least twice the computation time as our method.

6 Security of the confirmation protocol

In this section we discuss the security of the confirmation and denial protocols. A discussion of security against existential forgery is given in the next section.

Theorem 3. *The confirmation protocol has the following properties:*

Completeness: *If (m, s) has been formed correctly then Bob will accept Alice's proof.*

Soundness: *If (m, s) is not valid, then Alice, even computationally unbounded, cannot convince Bob to accept her proof with probability better than $1/B$.*

Zero-knowledge: *When Alice behaves correctly, Bob gains no information about Alice's private input apart from the validity of her proof.*

Proof. The proof of completeness is immediate from inspection of the protocol. We focus on the further two properties.

Soundness: We assume that $N = p_1 p_2$ has the property that $p_i \equiv 3 \pmod 4$ and that all odd primes $l < B$ are coprime to $\varphi(N)$. We also assume that the g_i generate Q_N. This is certified with probability $1 - 2^{-K_1}$ by the key certification process.

Let (m, s') be an invalid signature prepared by a cheating Alice. Then we can write $s' = \alpha \xi m^d \pmod N$ where $\xi^2 \equiv 1 \pmod N$ and where α has odd order which is divisible only by powers of primes q where $q \geq B$. Let q be such a prime. Since the g_i generate Q_N, there exists at least one index I such that $q | \mathrm{ord}(g_I)$.

Given a challenge C of the correct form ${s'}^{r_0} h_1^{r_1} \cdots h_k^{r_k}$ a cheating Alice must construct a response R which will satisfy Bob's check. We now project all elements into the subgroup $\langle \alpha \rangle$ of q elements which is generated by α (if $q \| \varphi(N)$ then this is done by raising all elements to the power $\varphi(N)/q$). We continue to use the same notation for these elements, but the reader should be aware that they now only have order dividing q.

Expressing all elements in terms of powers of α we have $m = \alpha^{l_0}$ and $g_i = \alpha^{l_i}$. It follows that

$$\log_\alpha(C) \equiv r_0(1 + l_0 d) + \sum_{i=1}^{k} l_i r_i d \pmod q \tag{1}$$

where $d \not\equiv 0 \pmod q$ and $l_I \not\equiv 0 \pmod q$ (where I is as above). The response R (again, only considering the image in the subgroup of elements of order q) must satisfy

$$\log_\alpha(R) \equiv r_0 l_0 + \sum_{i=1}^{k} l_i r_i \pmod q.$$

Since Alice knows d, the ability to construct the right response R is therefore equivalent to knowledge of $r_0 \pmod q$. But this is information-theoretically hidden: Given any solution $(r_0, \ldots, r_k)$ to equation (1) and any integer $0 < x < q$, there is another solution $(r_0 + x, r_1, \ldots, r'_I, \ldots, r_k)$ where

$$r'_I = (r_I l_I d - x(1 + l_0 d))/(l_I d) \pmod q.$$

Furthermore, since all values r_i may be reduced modulo $\varphi(N)$ the condition $1 < r_i < N$ is preserved. Hence, Alice has no better strategy than guessing the right power of α in her response R.

This argument applies to all primes $q | \mathrm{ord}(\alpha)$ but since Alice might choose α so that its order is the first prime power larger than B we can only conclude that Alice's cheating probability is $1/B$.

Zero-Knowledge: We analyse the protocol in the case where the commitment scheme is used. In this case Alice verifies the construction of C before giving Bob any information. Standard techniques show that the protocol can be simulated.

There is an interesting subtlety here though: If Alice's response R is actually of the form ξC^d then taking ratios gives an element ξ of order two, which is not simulatable. So only an "honest" run of the protocol is simulatable. In other words, if Alice at any time publishes signatures which are of the form ξm^d with $\xi \neq 1$ then she is potentially giving Bob useful information. □

7 Security against existential forgery

We now turn to the problem of whether an adversary can construct a pair (m, s) which Alice is unable to deny.

As is well known, with standard RSA [27] it is easy to forge signatures: Given (N, e) one can choose any integer $s \in \mathbb{Z}_N^*$, define $m = s^e \pmod{N}$, and then s is a valid signature for the message m. Similarly, for our system it is very easy to construct random pairs (m, s) such that $s \equiv m^d \pmod{N}$ using the pairs g_i, h_i. Hence resistance to forgery must rely heavily on the padding scheme.

Methods to form secure RSA encryption/signature schemes are one of the most important and well-known parts of cryptography. The precise techniques used to achieve this are not relevant to the present paper. We simply assume that some padding scheme such as that developed by Bellare and Rogaway [1] is used.

The proof of security against existential forgery given by Gennaro, Krawczyk and Rabin in [18] applies directly to our situation. The proof shows that an adversary who is able to forge a signature for the undeniable signature scheme can be used to construct an adversary which forges signatures for the same RSA modulus and the same message padding scheme. The security result applies to any attack model, in particular, it applies to an adversary mounting an adaptive chosen ciphertext attack (CCA2). The details of the message padding scheme do not affect the proof. We refer to [18] for the details of this, and also a discussion of indistinguishability of signatures.

8 Discussion

As with the scheme of Gennaro, Krawczyk and Rabin [18] it is possible to add extra functionality such as confirmability and convertibility. We discuss some of these extensions in this section. We also discuss the efficiency of the scheme.

8.1 Confirmability

Confirmer signatures are an extension of undeniable signatures which were introduced by Chaum [11]. These systems allow a signer to delegate the tasks of signature confirmation and denial to another party.

The secret key of Alice includes the number e which is used for verifying signatures. Alice may give this number to a designated confirmer. This means that the task of confirming signatures can be performed either by Alice or by the confirmer. Assuming the hardness of the RSA problem, since the confirmer only knows e, they are unable to forge Alice's signature.

We note that Alice cannot fool the confirmer about the validity of a given signature. To the confirmer, Alice's signatures are standard RSA signatures (up to multiplication by an element of small order).

8.2 Convertibility

More generally, Alice could have a private key of the form (e, d, c) where $edc \equiv 1 \pmod{\varphi(N)}$ In this case, e could be public, d and c known to Alice, and c known to a designated converter. The verification protocol in this case involves raising to the power ec rather than e and it is in all other respects identical to the one in Figure 3. An undeniable signature (m, s) can therefore be confirmed by the converter or the signer.

Any individual signature may be converted to a standard RSA signature by the transformation $(m, s) \to (m, s^c \pmod{N})$. This process may be performed by the converter or the signer. A proof analogous to that in Figure 3 must be used to show that this has been performed correctly. Once again, Alice cannot cheat against the confirmer.

Note that Alice may choose several different pairs (c_i, d_i) if she wants to use several converters with disjoint jurisdiction with the same public key (N, e).

Finally, by publishing c, all Alice's signatures become standard RSA signatures with respect to the exponent ec.

8.3 Efficiency

Here we discuss the performance of our scheme for general values of B, K_1 and K_2.

The one-off costs of public key certification are dominated by two factors. There are $\frac{K_1}{\log_2 3}$ iterations of the protocol of Figure 1, each iteration requiring one exponentiation by Alice and one by the CA. As we have discussed, when B becomes large (and the moduli special), these computations become costly. There are also K_1 iterations of the protocol of Figure 2, each iteration requiring $2k$ exponentiations (where k was defined as a function of K_1 and B earlier) for the two participants. These computations become *less* costly as B increases. The final public key is $2k + 1$ times as long as the modulus N.

Each of the $\frac{K_2}{\log_2 B}$ rounds of the signature confirmation protocol (in the zero-knowledge version) requires $k+2$ exponentiations for Alice and $2(k+1)$ for Bob. As we have discussed above, the denial protocol requires an additional pair of exponentiations in total.

For the typical choices of $K_1 = K_2 = 100$ and $B = 2^{10}$, we can take $k = 11$. Then key certification requires a few thousand exponentiations. For signature confirmation (and denial), Bob needs to do around 240 exponentiations and Alice around 10 (disregarding the cost of calculating and checking commitments). This is eminently practical.

When B is small then our signature confirmation is not as efficient as the scheme of [18]. On the other hand, the CA and the users can agree on how large B should be taken. Increasing the size of B reduces the size of k and improves the cost of signature confirmation at the expense of public key certification. For large B, say $B \geq 2^{20}$, one would use different certification techniques, e.g. the methods of [4]. The 'limiting case' is the case of [18], where signature confirmation is very efficient and public key certification is extremely inefficient.

We re-iterate that the scheme of [18] has improved efficiency when our signature denial protocol is used.

8.4 Completely general moduli

We assumed that $N = p_1 p_2$ (or, more generally, $N = p_1^{s_1} p_2^{s_2}$) where $p_i \equiv 3 \pmod 4$. One can construct a scheme which does not require any condition on $p_i \pmod 4$ but this involves some extra techniques. The main difficulty in this case is that we no longer know which power of two divides the exponent of $\mathbb{Z}_N^*$.

First, we assume that $\varphi(N)$ has at least one prime factor of sufficiently large size. If this is not the case then the modulus is easily factored using the Pollard $P-1$ method. No RSA-based cryptosystem would be secure with such a modulus.

The public key certification proceeds as before with some choice of B. For completely general moduli choose $B = 3$ (in which case it is not necessary to execute the protocol of Figure 1).

Let $M = 2^{\lfloor \log_2(N) \rfloor}$. A signature on a message m is now defined to be any element of the set

$$\{\xi m^d \in \mathbb{Z}_N^* : \xi^M \equiv 1 \pmod N\}.$$

The signature confirmation protocol proceeds as in Figure 3 except the check is that

$$R^M \equiv (m^{r_0} g_1^{r_1} \cdots g_k^{r_k})^M \pmod N.$$

The probability of successful cheating by a dishonest signer in one instance of the protocol is still $1/B$. The proof of security against forgery of signatures is a slight modification of the one given in [18].

8.5 Another approach

The trick used in the above subsection can be adapted to give a different solution to our original problem. Let $M = \prod_{2 \leq l < B} l^{\lfloor \log_l(N) \rfloor}$. Then a signature on message m could be any element such that $s^{eM} \equiv m^M \pmod N$. The advantage of this approach is that it can be used with a completely general modulus and no certification of the structure of the modulus is required.

There are two disadvantages of this approach. First, the size of the number M grows very quickly, and so only quite small values of B may realistically be used (not more than $B = 2^{10}$). Second, the size of candidate signatures could become extremely large. This has repercussions when analysing signature forgery, since for certain 'weak' moduli the probability of successful forgery might be relatively large compared to the desired security. Nevertheless, we feel it is worth mentioning this other approach.

9 Conclusion

We have presented an undeniable signature scheme for completely general RSA moduli. Our work provides a tunable family of schemes described by the parameters B, K_1 and K_2. These determine a set of trade-offs between efficiency of key certification (and generality of moduli) and the efficiency of signature confirmation/denial. For typical values, $B = 2^{10}$, $K_1 = K_2 = 100$, our scheme is completely practical. We also give a natural denial protocol for the RSA setting which is more efficient than the previous denial protocol of [18].

Acknowledgments

The authors would like to thank Ivan Damgård, Jan Camenisch and Markus Jakobsson for helpful comments. The first author would like to thank Hewlett-Packard Laboratories Bristol for support during the time this research was done.

References

1. Bellare, M. and Rogaway, P. The exact security of digital signatures–How to sign RSA and Rabin, in U. Maurer (ed.), EUROCRYPT '96, Springer LNCS 1070 (1996) 399–416.
2. Boyar, J., Chaum, C., Damgård, I. and Pedersen, T. Convertible undeniable signatures, in A.J. Menezes and S.A. Vanstone (eds.), CRYPTO '90, Springer LNCS 537, Springer (1991) 189–205.
3. Boyd, C. and Foo, E. Off-line fair payment protocols using convertible signatures, in K. Ohta et al (eds.), ASIACRYPT '98, Springer LNCS 1514 (1998) 271-285.
4. Camenisch J. and Michels, M. Proving in zero-knowledge that a number is the product of two safe primes, in J. Stern (ed.), EUROCRYPT '99, Springer LNCS 1592 (1999) 106–121.
5. Camenisch, J. and Michels, M. Confirmer signature schemes secure against adaptive adversaries, in B. Preneel (ed.), EUROCRYPT 2000, Springer LNCS 1870 (2000) 243-258.
6. Catalano, D., Gennaro, R. and Halevi, S. Computing inverses over a shared secret modulus, in B. Preneel (ed.), EUROCRYPT 2000, Springer LNCS 1807 (2000) 190–206.
7. Chaum, D. and van Antwerpen, H. Undeniable signatures, in G. Brassard (ed.), CRYPTO '89, Springer LNCS 435 (1990) 212-216.
8. Chaum, D. Zero-knowledge undeniable signatures, in I.B. Damgård (ed.), CRYPTO '90, Springer LNCS 473 (1991) 458-464.
9. Chaum, D., van Heijst, E., and Pfitzmann, B, Cryptographically strong undeniable signatures, unconditionally secure for the signer, in J. Feigenbaum (ed.), CRYPTO '91, Springer LNCS 576, (1992) 470–484.

10. Chaum, D. and Pedersen, T. P. Wallet databases with observers, in E. Brickell (ed.), CRYPTO '92, Springer LNCS 740 (1993) 89–105.
11. Chaum, D. Designated confirmer signatures, in A. de Santis (ed.), EUROCRYPT '94, Springer LNCS 950 (1995) 86–91.
12. Damgård, I. and Pedersen, T. New convertible undeniable signature schemes, in U. Maurer (ed.) EUROCRYPT '96, Springer LNCS 1070 (1996) 372–386.
13. Damgård, I. and Koprowski, M. Practical threshold RSA signatures without a trusted dealer, in B. Pfitzmann (ed.), EUROCRYPT 2001, Springer LNCS 2045 (2001) 152–165.
14. Desmedt, Y., Frankel Y. and Yung, M. Multi-receiver/multi-sender network security: efficient authenticated multicast/feedback, INFOCOM '92 (1992) 2045–2054.
15. Fouque, P.-A., Stern, J. Fully distributed threshold RSA under standard assumptions, in C. Boyd (ed.), ASIACRYPT 2001, Springer (2001).
16. Gennaro, R., Jerecki, S., Krawczyk, H. and Rabin, T. Robust and efficient sharing of RSA functions, in N. Koblitz (ed.), CRYPTO '96, Springer LNCS 1109 (1996) 157–172.
17. Gennaro, R., Miccianicio, D. and Rabin, T. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products, 5th ACM Conference on Computer and Communications Security, October 1998.
18. Gennaro, R., Krawczyk, H. and Rabin, T. RSA-based undeniable signatures, in W. Fumy (ed.), CRYPTO '97, Springer LNCS 1294 (1997) 132-149. Full version *Journal of Cryptology* (2000) 13:397–416.
19. van de Graaf, J. and Peralta, R. A simple and secure way to show that validity of your public key, in C. Pomerance (ed.), CRYPTO '87, Springer LNCS 293 (1988) 128–134.
20. Jakobsson, M., Sako, K. and Impagliazzo, R. Designated verifier proofs and their applications, in U. Maurer (ed.) EUROCRYPT '96, Springer LNCS 1070 (1996) 143–154.
21. Jakobsson, M. Efficient oblivious proofs of correct exponentiation, in B. Preneel (ed.), Communications and multimedia security, Kluwer (1999) 71–84.
22. Menezes, A.J., van Oorschot, P.C. and Vanstone, S.A. *Handbook of Applied Cryptography*, CRC Press, 1997.
23. Michels, M. and Stadler, M. Generic constructions for secure and efficient confirmer signature schemes, in K. Nyberg (ed.) EUROCRYPT '98, Springer LNCS 1403 (1998) 406–421.
24. Miyazaki, T. An improved scheme of the Gennaro-Krawczyk-Rabin undeniable signature system based on RSA, in D. Won (ed.), ICISC 2000, Springer LNCS 2015 (2001) 135–149.
25. Poupard, G. and Stern, J. Short proofs of knowledge for factoring, in PKC 2000, Springer LNCS 1751 (2000) 147–166.
26. Rabin., T. A simplified approach to threshold and proactive RSA, in H. Krawczyk (ed.), CRYPTO '98, Springer LNCS 1462 (1998) 89–104.
27. Rivest, R.L., Shamir, A. and Adleman, L.M. A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* v.21, n.2, 1978, pages 120–126.
28. Shoup, V. Practical threshold signatures, in B. Preneel (ed.), EUROCRYPT 2000, Springer LNCS 1807 (2000) 207–220.
29. Silverman, R. D. Fast generation of random, strong RSA primes, *CryptoBytes*, 3, No. 1, 1997, pp. 9–13.