A Signature Scheme with Efficient Protocols

Jan Camenisch
IBM Research
Zurich Research Laboratory
CH–8803 Rüschlikon
jca@zurich.ibm.com

Anna Lysyanskaya
Computer Science Department
Brown University
Providence, RI 02912 USA
anna@cs.brown.edu

Abstract

Digital signature schemes are a fundamental cryptographic primitive, of use both in its own right, and as a building block in cryptographic protocol design. In this paper, we propose a practical and provably secure signature scheme and show protocols (1) for issuing a signature on a committed value (so the signer has no information about the signed value), and (2) for proving knowledge of a signature on a committed value. This signature scheme and corresponding protocols are a building block for the design of anonymity-enhancing cryptographic systems, such as electronic cash, group signatures, and anonymous credential systems. The security of our signature scheme and protocols relies on the Strong RSA assumption. These results are a generalization of the anonymous credential system of Camenisch and Lysyanskaya.

1 Introduction

Digital signature schemes are a fundamental cryptographic application, invented together with public-key cryptography by Diffie and Hellman [16] and first constructed by Rivest, Shamir and Adleman [27]. They give the electronic equivalent of the paper-based idea of having a document signed.

A digital signature scheme consists of (1) a key generation algorithm that generates a public key PK and a secret key SK; (2) a signing algorithm that takes as inputs a message m and a secret key SK and generates a signature σ; and (3) a verification algorithm that tests whether some string σ is a signature on message m under the public key PK. Signature scheme exists if and only if one-way functions exist [25, 28]. However, the efficiency of these general constructions, and also the fact that these signature schemes require the signer's secret key to change between invocations of the signing algorithm, makes these solutions undesirable in practice.

Using an ideal random function (this is the so-called *random-oracle* model), several, much more efficient signature schemes were shown to be secure. Most notably, those are the RSA [27], the Fiat-Shamir [17], and the Schnorr [29] signature schemes. However, ideal random functions cannot be implemented in the plain model [12], and therefore in the plain model, these signature schemes are not provably secure.

Gennaro, Halevi, and Rabin [20] and Cramer and Shoup [13] proposed the first signature schemes whose efficiency is suitable for practical use and whose security analysis does not assume an ideal random function. Their schemes are secure under the so-called *Strong RSA assumption*.

In contrast to these stand-alone solutions, our goal is to construct signature schemes that are suitable as building blocks for other applications.

Consider the use of signature schemes for constructing an anonymous credential system. While such a system can be constructed from any signature scheme using general techniques for cryptographic protocol design, we observe that doing it in this fashion is very inefficient. Let us explain this point in more detail.

In a credential system, a user can obtain access to a resource only by presenting a credential that demonstrates that he is authorised to do so. In the paper-based world, examples of such credentials are passports that allow us to prove citizenship and authorise us to vote, driver's licenses that prove our ability to drive cars, etc. In the digital world, it is reasonable to imagine that such a credential will be in the form of a digital signature. Let us imagine that the user's identity is his secret key SK. Let us think of a credential as a signature on this secret key.

A credential system is anonymous if it allows users to demonstrate such credentials without revealing any additional information about their identity. In essence, when the user shows up before the verifier and demonstrates that he has a credential, the verifier can infer nothing about who the user is other than that the user has the right credential. Additionally, an anonymous credential system allows the user to obtain a credential anonymously.

Using general zero-knowledge proofs, it is possible to prove statements such as "I have a signature," without saying anything more than that (i.e., without disclosing what this credential looks like). However, doing so requires that the problem at hand be represented as, for example, a Boolean circuit, and then the proof that the statement is true requires a proof that the circuit has a satisfying assignment. This method, however, requires expensive computations beyond what is considered practical.

An additional complication is obtaining credentials in a secure way. The simple solution where the user reveals his identity SK to the credential granting organization, who in turn grants him the credential, is ruled out: we want to allow the user to be anonymous when obtaining credentials as well; we also want to protect the user from identity theft, and so the user's secret key SK should never be leaked to any other party. Here, general techniques of secure two-party computation save the day: the user and the organization can use a secure two-party protocol such that the user's output is a signature on his identity, while the organization learns nothing. But this is also very expensive: general secure two-party computation also represents the function to be computed as a Boolean circuit, and then proceeds to evaluate it gate by gate.

So far, we have come up with a construction of an anonymous credential system from a signature scheme using zero-knowledge proofs of knowledge and general two-party computation. We have also observed that this solution is not satisfactory as far as efficiency is concerned. A natural question therefore is whether we can create a signature scheme which will easily yield itself to an efficient construction of an anonymous credential system. In other words, we need signature schemes for which proving a statement such as "I have a signature," is a much more efficient operation than representing this statement as a circuit and proving something about such a circuit. We also need to be able to have the user obtain a signature on his identity without compromising his identity.

In this paper, we construct a signature scheme that meets these very general requirements. The generality of our approach enables the use of this signature scheme for other cryptographic protocols in which it is desirable to prove statements of the form "I have a signature," and to obtain signatures on committed values.

Our signature scheme is based on the Strong RSA assumption, and of all provably secure signatures known, it runs a close second in efficiency to the signature scheme of Cramer and Shoup [13].

Our results can be thought of as a generalization of the anonymous credential system due to Camenisch and Lysyanskaya [7], and can also be viewed as a generalization of the Ateniese et al. [1] group signature scheme.

In Section 2 we give the basic signature scheme, for which we give a proof of security in Section 3. In Section 4 we generalize this signature scheme to a scheme that allows to sign a block of messages instead of one message. Such a signature scheme can for example be used to encode public-key attributes as proposed by Brands [5], or to issue signatures that can only be used once (see Section 6.3). In section 5 we mention known protocols that are needed for our purposes, and then in section 6 we give protocols for signing a committed value and for proving knowledge of a signature.

To be more accessible to a non-specialist reader, our exposition also includes two extra sections. In the Appendix A, we explain our notation, and give the definition of a secure signature scheme, and a definition of a proof of knowledge. We also cover a few number-theoretic basics in Appendix B.

2 The Basic Signature Scheme

2.1 Preliminaries

Let us recall some basic definitions (see Appendix B for more in-depth discussion)

Definition 2.1 (Safe primes). *A prime number p is called* safe *if $p = 2p' + 1$, such that p' is also a prime number. (The corresponding number p' is known as a* Sophie Germain *prime.)*

Definition 2.2 (Special RSA modulus). *An RSA modulus $n = pq$ is* special *if $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes.*

Notation: By $QR_n \subseteq \mathbb{Z}_n^*$ we will denote the set of quadratic residues modulo n, i.e., elements $a \in \mathbb{Z}_n^*$ such that $\exists b \in \mathbb{Z}_n^*$ such that $b^2 \equiv a \bmod n$.

2.2 The Scheme

Key generation. On input 1^k, choose a special RSA modulus $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$ of the length $\ell_n = 2k$. Choose, uniformly at random, $a, b, c \in QR_n$. Output $PK = (n, a, b, c)$, and $SK = p$.

Message space. Let ℓ_m be a parameter. The message space consists of all binary strings of length ℓ_m. Equivalently, it can be thought of as consisting of integers in the range $[0, 2^{\ell_m})$.

Signing algorithm. On input m, choose a random prime number $e > 2^{\ell_m+1}$ of length $\ell_e = \ell_m + 2$, and a random number s of length $\ell_s = \ell_n + \ell_m + l$, where l is a security parameter. Compute the value v such that

$$v^e \equiv a^m b^s c \bmod n$$

Verification algorithm. To verify that the tuple (e, s, v) is a signature on message m in the message space, check that $v^e \equiv a^m b^s c \bmod n$, and check that $2^{\ell_e} > e > 2^{\ell_e - 1}$.

Efficiency analysis. In practice, $\ell_n = 1024$ is considered secure. As for the message space, if the signature scheme is intended to be used directly for signing messages, then $\ell_m = 160$ is good enough, since, given a suitable collision-resistant hash function, such as SHA-1, one can first hash a message to 160 bits, and then sign the resulting value. Then $\ell_e = 162$. The parameter l guides the statistical closeness of the simulated distribution to the actual distribution, hence in practice, $l = 160$ is sufficient, and therefore $\ell_s = 1024 + 160 + 160 = 1346$.

Therefore, this signature scheme requires one short (160-bit) and two long (1024 and 1346) exponentiations for signing, while verification requires two short (162 and 160 bits) and one long (1346-bit) exponentiations. This is not as efficient as the Cramer-Shoup signature, which requires three short (i.e., 160-bit) and one long (i.e., 1024-bit) exponentiations for signing, and four short exponentiations for verifying. However, our signature scheme has other redeeming qualities.

3 Proof of Security

We will show that the signature scheme in Section 2 is secure. Note that a forgery (m, s, e, v) may have value e of length ℓ_e not necessary a prime number.

We prove security under the Strong RSA assumption [2, 18]:

Assumption 3.1 (Strong RSA Assumption). *The strong RSA assumption states that it is hard, on input an RSA modulus n and an element $u \in \mathbb{Z}_n^*$, to compute values $e > 1$ and v such that $v^e \equiv u \bmod n$. More formally, we assume that for all polynomial-time circuit families $\{\mathcal{A}_k\}$, there exists a negligible function $\nu(k)$ such that*

$$\Pr[n \leftarrow \mathtt{RSAmodulus}(1^k); u \leftarrow QR_n; (v, e) \leftarrow \mathcal{A}_k(n, u) \ : e > 1 \ \wedge \ v^e \equiv u \bmod n] = \nu(k)$$

The tuple (n, u) generated as above, is called an *instance* of the *flexible* RSA problem.

The following lemma (proved in Appendix B, Lemma B.10) is important for the proof of security:

Lemma 3.1. *Let a special RSA modulus $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$, be given. Suppose we are given the values $u, v \in QR_n$ and $x, y \in Z$, $\gcd(x, y) < x$ such that $v^x \equiv u^y \bmod n$. Values z, $w > 1$ such that $z^w \equiv u \bmod n$ can be computed efficiently.*

Suppose that a forger's algorithm $\mathcal{F}$ is given. Suppose $\mathcal{F}$ makes K signature queries in expectation before it outputs a successful forgery. By the Markov inequality, half the time, $\mathcal{F}$ needs $2K$ or fewer queries. Without loss of generality, we may assume that K is known (since if it is not, then it can be estimated experimentally).

Suppose that the output of $\mathcal{F}$ is (m, s, e, v). Let us distinguish three types of outputs.

- *Type 1:* The forger's exponent e is relatively prime to all the exponents output by the signature oracle.
- *Type 2:* The forger's value e is not relatively prime to some previously seen exponent e_i and the root v is different from the corresponding root v_i.
- *Type 3:* The forger's values e is not relatively prime to some previously seen exponent e_i and $v = v_i$, but the tuple (m, s) is new.

By $\mathcal{F}_1$ (resp., $\mathcal{F}_2$, $\mathcal{F}_3$) let us denote the forger who runs $\mathcal{F}$ but then only outputs the forgery if it is of Type 1 (resp., Type 2, Type 3). The following lemma is clear by the standard hybrid argument:

Lemma 3.2. *If the forger $\mathcal{F}$ success probability is ϵ, then one of $\mathcal{F}_1$, $\mathcal{F}_2$, or $\mathcal{F}_3$ succeeds with probability at least $\epsilon/3$.*

Next, we will show that under the strong RSA assumption, success probabilities for each of $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$ are negligible.

Lemma 3.3. *Under the Strong RSA assumption, the success probability of $\mathcal{F}_1$ is negligible. More strongly, if $\mathcal{F}_1$ succeeds with probability ϵ_1 in expected time $O(p_1(k))$, using expected number $Q(k)$ queries, then there is an algorithm that runs in time $O(p_1(k))$ and solves the flexible RSA problem with probability $\epsilon/4$.*

Proof. Suppose $\mathcal{F}_1$ succeeds with non-negligible probability. Let us show that this contradicts the Strong RSA assumption.

Recall that Q is the expected number of queries, and by the Markov inequality, half the time, $\mathcal{F}_1$ will need at most $2Q = O(p_1(k))$ queries.

Let us construct an algorithm $\mathcal{A}$ such that, on input a strong RSA instance (n, u), and with access to $\mathcal{F}_1$, $\mathcal{A}$ will output (v, e) such that $e > 1$ and $v^e \equiv u \bmod n$.

Now $\mathcal{A}$ plays the role of the signer, as follows:

Key generation: Choose $2Q$ prime numbers $e_1, \ldots, e_{2Q}$. Choose, at random, value $r_1 \in \mathbb{Z}_{n^2}$ and $r_2 \in \mathbb{Z}_{n^2}$. Let $a = u^E$, where $E = \prod_{i=1}^{2Q} e_i$, $b = a^{r_1}$, $c = a^{r_2}$. Let (n, a, b, c) be the public key.

Signing: Upon receiving the i^{th} signature query m_i, choose a value s_i of length ℓ_s uniformly at random. Output (e_i, s_i, v_i), where $v_i = a_i^{m_i} b_i^{s_i} c_i$, where $a_i = u^{E/e_i}$, $b_i = a_i^{r_1}$, $c_i = a_i^{r_2}$. (In other words, a_i, b_i and c_i are values such that $a_i^{e_i} = a$, $b_i^{e_i} = b$, and $c_i^{e_i} = c$). Note that (e_i, s_i, v_i) constitute a valid signature:

$$\begin{aligned} v_i^{e_i} &= (a_i^{m_i} b_i^{s_i} c_i)^{e_i} \\ &= a^{m_i} b^{s_i} c \ . \end{aligned}$$

Note that the public key and all the signatures have the same distribution as in the original scheme.

Suppose the forgery is (e, s, v) on message m, and $e > 4$. This gives us $v^e = a^m b^s c = u^{E(m+r_1 s+r_2)}$. Observe that if it is the case that e and $E(m + r_1 s + r_2)$ are relatively prime, then by Shamir's trick (Lemma B.9) we are done. We will not necessarily have this nice case, but we will show that things will be good enough.

First, by assumption on $\mathcal{F}_1$ we have $\gcd(e, E) = 1$. Second, we will use the following claim:

Claim. *With probability at least $1/2$ over the random choices made by the reduction, it is the case that either $\gcd(e, m + r_1 s + r_2) < e$, or, on input e, one can efficiently factor n.*

The claim immediately gives us the conditions of Lemma 3.1, which gives us values v, $w > 1$ such that $v^w \equiv u \bmod n$.

We must now prove the claim. Suppose that it is false. Then the probability that $\gcd(e, m+r_1 s+r_2) < e$ is less than $1/2$. Suppose $r_2 = x+\phi(n)z$. Note that z is independent of the view of the adversary and it is chosen statistically indistinguishable from random over $\mathbb{Z}_n$. Therefore, if with probability greater than $1/2$, $e \mid m + r_1 s + r_2$, this holds for greater than $1/2$ of all the possible choices for z. Then there exists a value $z_0 \in \mathbb{Z}_n$ such that $e \mid (m + r_1 s + x + \phi(n)z_0)$ and $e \mid (m + r_1 s + x + \phi(m)(z_0 + 1))$. Then it holds that $e \mid \phi(n)$ with probability at least $1/2$. Therefore, with probability $1/2$, we either factor n (by Lemma B.5) or $\gamma = \gcd(e, m + r_1 s + r_2) < e$.

Therefore, $\mathcal{A}$ succeeds in breaking the strong RSA assumption whenever the number of queries does not exceed $2Q$ (which happens with probability $1/2$ by Markov inequality) and whenever the conditions of the claim are satisfied (which happens with probability $1/2$, independently on the number of queries), and so the success probability of $\mathcal{A}$ is $\epsilon_1/4$, while its running time is $O(p_1(k))$. □

Lemma 3.4. *Under the Strong RSA assumption, the success probability of $\mathcal{F}_3$ is negligible. More strongly, if $\mathcal{F}_3$ succeeds with probability ϵ_1 in expected time $O(p_1(k))$, using expected number $Q(k)$ queries, then there is an algorithm that runs in time $O(p_3(k))$ and succeeds with probability $\epsilon_3/2$.*

Proof. The input to the reduction is (n, e), an instance of the flexible RSA problem.

The key generation and signatures in this case are as in the proof of Lemma 3.3. The signatures (m, s, e, v) and (m_i, s_i, e_i, v) give us the following: Note that from $\gcd(e_i, e) \neq 1$ and the fact that $2^{\ell_e} > e, e_i > 2^{\ell_e}$ it follows that $e_i = e$ and thus $a^m b^s \equiv a^{m_i} b^{s_i}$. This implies that $m + r_1 s \equiv m_i + r_1 s_i \bmod \phi(n)$. Since $(m, s) \neq (m_i, s_i)$, and $r_1 > m$, $r_1 > m_i$,

$m + r_1 s \neq m_i + r_1 s_i$. Therefore, $\phi(n) \mid m + r_1 s - m_i + r_1 s_i \neq 0$, and so by Corollary B.6, we are done.

The probability that $\mathcal{A}$ succeeds in breaking the Strong RSA assumption is at least $\epsilon/2$ because with probability at least $1/2$, the forger $\mathcal{F}_3$ will need at most $2Q$ queries. $\square$

Lemma 3.5. *Under the Strong RSA assumption, the success probability of $\mathcal{F}_2$ is negligible. More strongly, if $\mathcal{F}_2$ succeeds with probability ϵ_2 in expected time $O(p_2(k))$, using expected number $Q(k)$ queries, then there is an algorithm that runs in time $O(p_2(k))$ and succeeds with probability $\epsilon(k)/8Q(k)$.*

Proof. The input to the reduction is (n, u), an instance of the flexible RSA problem.

Choose a value $1 \leq i \leq 2Q(k)$ uniformly at random. With probability $1/2Q(k)$, the exponent e will be the same as in the i'th query to the signature oracle.

The idea of this reduction is that we will set up the public key in such a way that, without knowledge of the factorization of n, it is possible to issue correctly distributed signatures on query $j \neq i$, in almost the same manner as in the proof of Lemma 3.3. For query i, the answer will be possible only for a specific, precomputed value of $t = m\alpha + s$, where $\alpha = \log_b a$.

Key Generation: Choose $2Q(k)$ random primes $\{e_j\}$. Let $E = (\prod_{j=1}^{2Q} e_j)/e_i$. Choose $\alpha, \beta \in \mathbb{Z}_{2n}$ uniformly at random. Choose a random value t of length ℓ_s. Let $b = u^E \bmod n$. Let $a = b^\alpha \bmod n$, and $c = b^{e_i\beta - t}$. Let (n, a, b, c) be the public key.

Signing: Upon receiving the j^{th} signature query m_j, $j \neq i$, choose a value s_j of length ℓ_s uniformly at random. Output (e_j, s_j, v_j), where $v_j = a_j^{m_j} b_j^{s_j} c_j$, where $b_j = u^{E/e_j}$, $a_j = b_j^\alpha$, $c_j = b_j^{e_i\beta - t}$. (In other words, a_j, b_j and c_j are values such that $a_j^{e_j} = a$, $b_j^{e_j} = b$, and $c_j^{e_j} = c$). Note that (e_j, s_j, v_j) constitute a valid signature.

Upon receiving the i^{th} signature query m_i, compute the value $s_i = t - \alpha m_i$. $v_i = b^\beta$. Note that $v_i^{e_i} = b^{e_i\beta - t + t} = b^t c = a^{m_i} b^{s_i} c$. Note that if ℓ_s is sufficiently long (e.g., $\ell_n + \ell_m + l$, where ℓ_n is the length of the modulus n, ℓ_m is the length of a message, and l is a security parameter), s_i is distributed statistically close to uniform over all strings of length ℓ_s.

Suppose the forgery gives (m, s, e, v) such that e is not relatively prime to e_i. Then $e = e_i$ because e is of length ℓ_e. Also, as the verification is successful for the forgery as well as for the i'th query, $v^{e_i} = a^m b^s c = b^{m\alpha + s + e_i\beta - t} = b^{(m-m_i)\alpha + (s-s_i) + e_i\beta}$. Let $\alpha = \alpha_1\phi(n) + \alpha_2$. Note that with probability very close to 1, the value $\alpha_1 \in \{0, 1\}$. Also, note that α_1 is independent on the adversary's view. Therefore, if the probability that $e_i \mid (m - m_i)\alpha + (s - s_i) + e_i\beta$ is non-negligibly higher than $1/2$, then it is the case that $e_i \mid (m - m_i)\alpha_2 + (s - s_i) + e_i\beta$ and $e_i \mid (m - m_i)(\phi(n) + \alpha_2) + (s - s_i) + e_i\beta$, and therefore $e_i \mid (m - m_i)$ (note that by definition of a forgery $m \neq m_i$). If the length ℓ_e is two bits longer than the length of a valid message ℓ_m, this is impossible.

Therefore, with probability at least $1/2$, the following condition is met: $e_i \nmid (m\alpha + s + e_i\beta - t) = \gamma$. Therefore, we have the following: $v^{e_i} \equiv u^{E\gamma}$, where $\gcd(e_i, \gamma) = 1$, and so by Lemma B.9, we compute e_i'th root of u.

Therefore, $\mathcal{A}$ succeeds in solving an instance of the flexible RSA problem when $2Q$ is a sufficient number of queries (with probability at least $1/2$ this is the case by the Markov inequality), when i is chosen correctly (this is the case with probability $1/2Q$) and when the condition is met (with probability $1/2$ independently of everything else). Therefore, $\mathcal{A}$ succeeds with probability $\epsilon_2/8Q$ in time $p_2(k)$. $\square$

Since the number of signature queries is at most the running time, we have shown the following theorem:

Theorem 3.6. *Our signature scheme is secure under the Strong RSA assumption. More precisely, if a forger breaks our signature scheme in time $p(k)$ with probability $\epsilon(k)$, then the Strong RSA assumption can be broken in time $O(p(k))$ with probability $\Omega(\epsilon(k)/p(k))$.*

4 The Scheme for Blocks of Messages

Suppose, instead of signing a single message m, one wants to sign a block of L messages $m_1, \ldots, m_L$. For most applications, these two problems are equivalent, since in order to sign such a block, it is sufficient to hash it to a small message m using a collision-resistant hash function, and then to sign m using a one-message signature scheme.

However, if the application we have in mind involves proving knowledge of a signature and proving relations among certain components of $m_1, \ldots, m_L$, it may be helpful to have a signature scheme especially designed to handle blocks of L messages. Here, we propose one such scheme.

Key generation. On input 1^k, choose a special RSA modulus $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$. Choose, uniformly at random, $a_1, \ldots, a_L, b, c \in QR_n$. Output $PK = (n, a_1, \ldots, a_L, b, c)$, and $SK = p$. Let ℓ_n denote the length of the modulus n. $\ell_n = 2k$.

Message space. Let ℓ_m be a parameter. The message space consists of all binary strings of length ℓ_m. Equivalently, it can be thought of as consisting of integers in the range $[0, 2^{\ell_m})$.

Signing algorithm. On input $m_1, \ldots, m_L$, choose a random prime number $e > 2^{\ell_m+1}$ of length $\ell_e = \ell_m + 2$, and a random number s of length $\ell_s = \ell_n + \ell_m + l$, where l is a security parameter. Compute the value v such that

$$v^e \equiv a_1^{m_1} \ldots a_L^{m_L} b^s c \bmod n$$

Verification algorithm. To verify that the tuple (e, s, v) is a signature on message m, check that $v^e \equiv a_1^{m_1} \ldots a_L^{m_L} b^s c \bmod n$, and check that $e > 2^{\ell_e - 1}$.

Theorem 4.1. *The signature scheme above is secure for blocks of messages, under the Strong RSA assumption.*

Proof. We will show a reduction from a forger $\mathcal{F}$ for this signature scheme, to an algorithm that breaks the basic signature scheme from Section 2. The reduction receives as input the public key of the basic scheme: (n, a, b, c), and has oracle access to the basic signature scheme.

Key Generation. Choose an index $I \in [1, L]$ at random. Let Let $a_I = a$. For $j \in [1, L]$, $j \neq I$, choose and integer r_j from $\mathbb{Z}_{2n}$ uniformly at random, and let $a_j = b^{r_j}$. Output the public key $(n, a_1, \ldots, a_L, b, c)$.

Signature generation. On input signature query $(m_1, \ldots, m_L)$, ask the signature oracle for a signature on the value m_I. Upon receiving the value (s, v, e), let $s' = s - \sum_{j \neq I} r_j m_j$. The resulting signature is (s', v, e). It is easy to see that it is distributed statistically close to the right distribution provided that $\ell_s = \ell_n + \ell_m + l$, where l is sufficiently long.

Suppose the adversary's forgery is $(m_1, \ldots, m_L, s, v, e)$. Then $(s + \sum_{j=2}^{L} r_j m_j, v, e)$ is a valid signature in the basic scheme on message m_I. Suppose the signature's e is different from any previously seen e_i. Then we are done: we have created forger $\mathcal{F}_\infty$ and contradicted Lemma 3.3. So suppose this $e = e_i$. But $(m_1, \ldots, m_L) \neq (m_1^i, \ldots, m_L^i)$. With probability at least $1/L$, it is the case that $m_I \leq m_I^i$, and we have created a forged signature on message m_I.

Overall, if the adversary $\mathcal{F}$ succeeds in time $p(k)$ with probability ϵ using $Q(k)$ queries, the reduction succeeds against the basic scheme in time $p(k)$ with probability at least $\epsilon/2L$ also using $Q(k)$ queries. $\square$

5 Preliminary Protocols

In this section, we will show how to construct a secure protocol for signing a committed message as described in Section 1, under our basic signature scheme described in Section 2.

5.1 Commitment Scheme

The following commitment scheme is due to Fujisaki and Okamoto [19] and elaborated on by Damgård and Fujisaki [15]. Its security relies on the hardness of factoring.

Key generation The public key consists of a special RSA modulus n of length ℓ_n, and $h \leftarrow QR_n$, $g \leftarrow \langle h \rangle$, where $\langle h \rangle$ is the group generated by h.

Commitment The commitment $\texttt{Commit}(PK, x, r)$ for inputs of length ℓ_x and randomness $r \in \mathbb{Z}_n$, is defined as follows: $\texttt{Commit}((n, g, h), x, r) = g^x h^r \bmod n$.

Lemma 5.1. *The commitment scheme described above is statistically hiding and computationally binding if factoring is hard.*

Remark. Note that for the hiding property, the only requirement from the public key is that $g \in \langle h \rangle$, and so the public key can be generated adversarially provided that this requirement holds.

We want to guarantee the security of this commitment scheme even when the commitment scheme's public key is provided by the adversary. To ensure that, we add the following key certification procedure: the party that generated the commitment keys (n, g, h) must prove existence of the discrete logarithm $\log_h g \bmod n$. This can be done using well-known techniques that we elaborate on in the sequel (cf. [15]).

5.2 Known Discrete-Logarithm-Based Protocols

All protocols assume that the public parameters were fixed after the adversary was fixed (otherwise a non-uniform adversary may know secret keys).

The following known protocols are secure under the strong RSA assumption:

- Proof of knowledge of discrete logarithm representation modulo a composite [18]. That is to say, a protocol with common inputs (1) a $(n, g_1, \ldots, g_m)$, where n is a special RSA modulus and $g_i \in QR_n$ for all $1 \leq i \leq m$; and (2) a value $C \in QR_n$, the prover proves knowledge of values $\alpha_1, \ldots, \alpha_m$ such that $C = \prod_{i=1}^{m} g_i^{\alpha_i} \bmod n$.

- Proof of knowledge of equality of representation modulo two (possibly different) composite moduli [9]. That is to say, a protocol with common inputs $(n_1, g_1, \ldots, g_m)$ and $(n_2, h_1, \ldots, h_m)$ and the values $C_1 \in QR_{n_1}$ and $C_2 \in QR_{n_2}$, the prover proves knowledge of values $\alpha_1, \ldots, \alpha_m$ such that

$$C_1 = \prod_{i=1}^{m} g_i^{\alpha_i} \bmod n_1 \qquad \text{and} \qquad C_2 = \prod_{i=1}^{m} h_i^{\alpha_i} \bmod n_2 \ .$$

- Proof that a committed value is the product of two other committed values [8]. That is to say, a protocol with common inputs (1) a commitment key (n, g, h) as described in Section 5.1; and (2) values C_a, C_b, C_{ab} in QR_n, where the prover proves knowledge of the integers α, β, ρ_1, ρ_2, ρ_3 such that $C_a = g^{\alpha} h^{\rho_1} \bmod n$, $C_b = g^{\beta} h^{\rho_2} \bmod n$, and $C_{ab} = g^{\alpha\beta} h^{\rho_3} \bmod n$.

- Proof that a committed value lies in a given integer interval [4]. That is to say, a protocol with common inputs (1) a commitment key (n, g, h) as described in Section 5.1; (2) a value $C \in QR_n$; (3) integers a and b, where the prover proves knowledge of the integers α and ρ such that $C = g^{\alpha} h^{\rho} \bmod n$ and $a \leq \alpha \leq b$.

Notation: In the sequel, we will sometimes use the notation introduced by Camenisch and Stadler[10] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha, \beta, \gamma) : y = g^{\alpha} h^{\beta} \ \wedge \ \tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\gamma} \ \wedge \ (u \leq \alpha \leq v)\}$$

denotes a "*zero-knowledge* P*roof of* K*nowledge of integers* α, β, *and* γ *such that* $y = g^\alpha h^\beta$ *and* $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma$ *holds, where* $v < \alpha < u$," where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote quantities the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, a proof-protocol can be described by just pointing out its aim while hiding all details.

6 Protocols for the Signature Scheme

In the sequel, we rely on honest-verifier zero-knowledge proofs of knowledge of representation protocols; they can be converted into a general zero-knowledge proof using standard techniques based on trapdoor commitment schemes (cf. [14]).

6.1 Protocol for Signing a Committed Value

Informally, in a protocol for signing a committed value, we have a signer with public key PK, and the corresponding secret key SK, and a user who queries the signer for a signature. The common input to the protocol is a commitment C for which the user knows the opening (m, r). In the end of the protocol, the user obtains a signature $\sigma_{PK}(m)$, and the signer learns nothing about m.

Common inputs: Public key (n, a, b, c) of the signature scheme with parameters ℓ_m, ℓ_s, ℓ_e.

Commitment public key (n_C, g_C, h_C), commitment C.

User's input: x and r_C such that $C = g_C^x h_C^{r_C} \bmod n_C$.

Signer's input: Factorization of n.

$U \longleftrightarrow S$ Form commitment $C_x = a^x b^r \bmod n$, and prove that C_x is a commitment to the same value as C.

Prove knowledge of x, r. Prove that $x \in (0, 2^{\ell_m})$, and $r \in (0, 2^{\ell_n})$.

$U \longleftarrow S$ Choose a random r' of length ℓ_s, and a prime number of length ℓ_e. Compute $v = (C_x b^{r'} c)^{1/e}$. Send (r', e, v) to the user.

User's output: Let $s = r + r'$. The user outputs a signature (s, e, v) on x.

Figure 1: Signature on a Committed Value

Lemma 6.1. *Figure 1 describes a secure protocol for signing a committed value.*

Proof. Let the extractor E be as follows: it runs the extractor for step 6.1 of the protocol, and discovers x and r that are of lengths ℓ_x and ℓ_r respectively. It then queries the signature oracle for a signature on x. It obtains the signature (s, e, v). Note that the value s is a random integer of length ℓ_s. The extractor then sets $r' = s - r$. If $\ell_s \geq \ell_n + \ell$, where ℓ is a security parameter guiding the statistical closeness, then the value r' obtained this way will be statistically close to the value r' generated by S, and so statistically, the adversary's view is the same whether talking to the real signer S, or to the extractor E. $\square$

6.2 Proof of Knowledge of a Signature

In this section, we give a zero-knowledge proof of knowledge protocol for showing that a committed values is a signature on another committed value.

Lemma 6.2. *The protocol in Figure 2 is a zero-knowledge proof of knowledge of the values* (x, r_x, s, e, v) *such that* $C_x = g^x h^{r_x} \bmod n$ *and* $\texttt{Verify}_{PK}(x, s, e, v)$.

Proof. The completeness property is clear. The zero-knowledge property follows because the simulator can form the commitments at random (since they reveal nothing statistically), and then invoke the simulator for the zero-knowledge proofs of knowledge of representation and belonging to an interval.

We must now show that there exists an extractor that computes a valid signature. Our extractor will invoke the extractor for the proof protocols we use as a building block. If our extractor fails, then we can set up a reduction that breaks the Strong RSA assumption.

Suppose the extractor succeeds and computes $(x, s, e, w, z, r_x, r_s, r_e, r_w, r_z, r)$ such that $C = (C_v)^e h^r$ and $C = a^x b^s c g^z h^r$, and $C_z = g^{ew} h^{r_z}$. Note that this implies that $C_v^e = a^x b^s c g^z$. Let $v = C_v / g^w$. Note that $v^e = C_v^e / g^z = a^x b^s c$. Since we also know that x is of length ℓ_m, s is of length ℓ_s and e is of length ℓ_e, we are done: we output (s, e, v) which is a signature on the message x. $\square$

6.3 Protocols for Signatures on Blocks of Messages

We note that straightforward modifications to the protocols above give us protocols for signing blocks of committed values, and to prove knowledge of a signature on blocks of committed values. We also note that, using any protocol for proving relations among components of a discrete-logarithm representations of a group element [11, 8, 5], can be used to demonstrate relations among components of a signed block of messages. We highlight this point by showing a protocol that allows an off-line double-spending test.

In order to enable an off-line double-spending test, a credential is a signature on a tuple (SK, N_1, N_2), and in a spending protocol (i.e., credential showing) a user reveals the value $b = aSK + N_1 \bmod q$, for some prime number q, and N_2, for a value $a \in \mathbb{Z}_q$ chosen by the verifier. The user must then prove that the value revealed corresponds to the signed block of messages (SK, N_1, N_2).

Here, we describe a mechanism that, combined with our signature scheme for blocks of messages, achieves this. We note that these types of techniques are similar to those due to Brands [5].

We assume that we are given, as public parameters, public values $(p = 2q + 1, g_p)$ where p and q are primes, and $g_p \mathbb{Z}_p$ has order q. We require that $q > \max(2^{\ell_m}, 2^{\ell_N})$ where ℓ_N is the length of the nonce N_1. We also assume that we are given another commitment public key, (n_C, g_C, h_C), same as in the rest of this Chapter.

In order to prove that the values (b, N_2) correspond to the signature on his secret key committed to in commitment C_{SK} the user forms commitments and C_1 to N_1 and C_2 to N_2. He then shows that he has a signature on the block of values committed to by this block of commitments, and that the value committed to in C_1 is of length ℓ_N. He reveals N_2 and shows that C_2 is a valid commitment to N_2. He also carries out the following proof of knowledge protocol:

$$PK\{(\alpha, \beta, r_\alpha, r_\beta) \ : \ g_p^b = (g_p^a)^\alpha g^\beta \ \wedge \ C_{SK} = g_C^\alpha h_C^{r_\alpha} \ \wedge \ C_1 = g_C^\beta h_C^{r_\beta} \ \} \ .$$

Lemma 6.3. *The protocol described above is a proof of knowledge of a block of values* (SK, N_1) *such that* (SK, N_1, N_2) *is signed, and* C_{SK} *is a commitment to* SK*, and* $b = aSK + N_1 \bmod q$.

Proof. (Sketch) The completeness and zero-knowledge properties follow in the usual way. We must now address the question of extraction. The fact that we extract (SK, N_1, N_2) and a signature on them follows similarly as for the protocol in Figure 2. The only thing we worry about is that $b = aSK + N_1 \bmod q$. Using a knowledge extractor from the proofs of knowledge of equal representations, we extract values α and β of lengths ℓ_m and ℓ_N such that $\log_{g_p}(g_p^b) = b = \alpha a + \beta \bmod q$, (since the order of g_p is q) and α is the value committed to in C_{SK} while β is the value committed to in C_1. Under the strong RSA assumption, it follows that $\alpha = SK$ and $\beta = N_1$, and so $b = SKa + N_1 \bmod q$ as we require. □

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer Verlag, 2000.

[2] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer Verlag, 1997.

[3] M. Bellare and O. Goldreich. On defining proofs of knowledge. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer-Verlag, 1992.

[4] F. Boudot. Efficient proofs that a committed number lies in an interval. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.

[5] S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy.* PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[6] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, Oct. 1988.

[7] J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Verlag, 2001.

[8] J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer Verlag, 1999.

[9] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430. Springer Verlag, 1999.

[10] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410–424. Springer Verlag, 1997.

[11] J. L. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* PhD thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.

[12] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–218, 1998.

[13] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proc. 6th ACM Conference on Computer and Communications Security*, pages 46–52. ACM press, nov 1999.

[14] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.

[15] I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. `http://eprint.iacr.org/2001`, 2001.

[16] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, IT-22(6):644–654, Nov. 1976.

[17] A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Verlag, 1987.

[18] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 1997.

[19] E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 32–46. Springer Verlag, 1998.

[20] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer Verlag, 1999.

[21] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal of Computing*, 18(1):186–208, Feb. 1989.

[22] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[23] S. Micali. 6.875: Introduction to cryptography. MIT course taught in Fall 1997.

[24] G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.

[25] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, Washington, 15–17 May 1989. ACM.

[26] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[27] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[28] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, Baltimore, Maryland, 1990. ACM.

[29] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

[30] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. In *ACM Transaction on Computer Systems*, volume 1, pages 38–44, 1983.

A On the Notation and Definitions Used

A.1 Notation

Let A be an algorithm. By $A(\cdot)$ we denote that A has one input (resp., by $A(\cdot,\ldots,\cdot)$ we denote that A has several inputs). By $A_{(\cdot)}$ we will denote that A is an indexed family of algorithms.

$y \leftarrow A(x)$ denotes that y was obtained by running A on input x. In case A is deterministic, then this y is unique; if A is probabilistic, then y is a random variable. If S is a set, then $y \leftarrow S$ denotes that y was chosen from S uniformly at random.

Let A and B be interactive Turing machines. By $(a \leftarrow A(\cdot) \leftrightarrow B(\cdot) \rightarrow b)$, we denote that a and b are random variables that correspond to the outputs of A and B as a result of their joint computation.

Let b be a boolean function. The notation $(y \leftarrow A(x) : b(y))$ denotes the event that $b(y)$ is true after y was generated by running A on input x.

Finally, the statement

$$\Pr[\{x_i \leftarrow A_i(y_i)\}_{1\le i\le n} \ : \ b(x_n)] = \alpha$$

means that the probability that $b(x_n)$ is TRUE after the value x_n was obtained by running algorithms $A_1, \ldots, A_n$ on inputs $y_1, \ldots, y_n$, is α.

(The above notation is from the Cryptography course taught by Silvio Micali at MIT [23].)

By $A^{(\cdot)}(\cdot)$, we denote a Turing machine that makes oracle queries. I.e., this machine will have an additional (read/write-once) query tape, on which it will write its queries in binary; once it is done writing a query, it inserts a special symbol "#". By external means, once the symbol "#" appears on the query tape, an oracle is invoked and its answer appears on the query tape adjacent to the "#" symbol. By

$$Q = Q(A^O(x)) \leftarrow A^O(x)$$

we denote the contents of the query tape once A terminates, with oracle O and input x. By $(q, a) \in Q$ we denote the event that q was a query issued by A, and a was the answer received from oracle O.

By $A^{1(\cdot)}(\cdot)$, we denote a Turing machine that makes just one oracle query. Analogously, by $A^{\ell(\cdot)}(\cdot)$, we denote a Turing machine that makes ℓ oracle queries.

By $A \overset{\blacksquare}{\to} B$ we denote that algorithm A has black-box access to algorithm B, i.e. it can invoke B with an arbitrary input and reset B's state.

We say that $\nu(k)$ is a negligible function, if for all polynomials $p(k)$, for all sufficiently large k, $\nu(k) < 1/p(k)$.

A.2 Signature Scheme

First, we cite the definition of a secure signature scheme.

Definition A.1 (Signature scheme [22]). *Probabilistic polynomial-time algorithms* $(G(\cdot), \mathtt{Sign}_{(\cdot)}(\cdot), \mathtt{Verify}_{(\cdot)}(\cdot,\cdot))$, *where G is the key generation algorithm,* `Sign` *is the signature algorithm, and* `Verify` *the verification algorithm, constitute a digital signature scheme for efficiently samplable (in the length of its index) message space $\mathcal{M}_{(\cdot)}$ if for all $m \in \mathcal{M}_{(\cdot)}$*

Correctness *If a message m is in the message space for a given PK, and SK is the corresponding secret key, then the output of $\mathtt{Sign}_{SK}(m)$ will always be accepted by the verification algorithm $\mathtt{Verify}_{PK}$. More formally, for all values m:*

$$\Pr[(PK, SK) \leftarrow G(1^k); \sigma \leftarrow \mathtt{Sign}_{SK}(m) \ : \ m \in \mathcal{M}_{PK} \wedge \neg\mathtt{Verify}_{PK}(m, \sigma')] = 0$$

Security *Even if an adversary has oracle access to the signing algorithm which provides signatures on messages of the adversary's choice, the adversary cannot create a valid signature on a message not explicitly queried. More formally, for all families of probabilistic polynomial-time oracle Turing machines $\{A_k^{(\cdot)}\}$, there exists a negligible function $\nu(k)$ such that*

$$\begin{aligned}\Pr[(PK,SK) \leftarrow G(1^k); (Q,x,\sigma) \leftarrow A_k^{\mathtt{Sign}_{SK}(\cdot)}(1^k) :&\\ \mathtt{Verify}_{PK}(m,\sigma) = 1 \wedge \neg(\exists \sigma' \mid (m,\sigma) \in Q)] \quad &= \quad \nu(k)\end{aligned}$$

A.3 Proof of Knowledge

We also give a definition of a zero-knowledge black-box proof of knowledge protocol. Zero knowledge proofs of knowledge were introduced independently by Brassard, Chaum, and Crépeau [6] and by Goldwasser, Micali, and Rackoff [21], and further refined by Bellare and Goldreich [3].

The definition we give here is both stronger and simpler than some other, more general definitions, for example the one due to Goldreich and Bellare. Since our main goal here is in exhibiting protocols, this is good enough since our protocols satisfy this stronger definition. In particular, our knowledge extraction property implies very strong soundness, and so some protocols that the literature considers to be proofs of knowledge would not be so under our definition.

Let x be an input, and let R be a polynomially computable relation. A zero-knowledge proof of knowledge of a witness w such that $R(x,w) = 1$ is a probabilistic polynomial-time protocol between a prover P and a verifier V such that there exists an expected polynomial-time simulator S and a polynomial-time extractor E such that:

1. Completeness: For all

$$\Pr[P(1^k,x,w) \leftrightarrow V(1^k,x) \rightarrow b \ : \ b = 1] = 1$$

2. Black-box zero-knowledge: For all $(x,w) \in R$, for all polynomial auxiliary inputs a, for all probabilistic polynomial-time families of Turing machines $\{V_k\}$, there exists a negligible function such that

$$\begin{aligned}|\Pr[P(1^k,x,w) \leftrightarrow V_k(a,x) \rightarrow b \ : \ b = 1] \quad &- \\ \Pr[S(1^k,x) \stackrel{\blacksquare}{\rightarrow} V_k(a,x) \rightarrow b \ : \ b = 1]| \quad &= \quad \nu(k)\end{aligned}$$

3. Black-box extraction: For all x, for all $\{P_k\}$, for all auxiliary inputs a, if for any function ϵ $\Pr[P_k(a,x) \leftrightarrow V(1^k,x) \rightarrow b \ : \ b = 1] \geq \epsilon(k)$, then there exists a constant c and a negligible function $\nu(k)$ such that $\Pr[P_k(a,x) \stackrel{\blacksquare}{\leftarrow} E(1^k,x) \rightarrow w \ : \ R(x,w) = 1] \geq \epsilon^c(k) - \nu(k)$. In other words, if the prover convinces the verifier often, then the extractor computes the witness almost as often. Note that since the relation R is polynomial-time computable, the extractor can always just test whether its output satisfies $(x,w) \in R$. Let (E) denote the extractor that runs E with black-box access to P_k until w is computed.

Definition A.2 (Proof of knowledge). *A protocol that satisfies the above description for relation R is called a proof of knowledge protocol for relation R.*

B Number-Theoretic Crash Course

Definition B.1 (RSA modulus). *A $2k$-bit number n is called an* RSA modulus *if $n = pq$, where p and q are k-bit prime numbers.*

Definition B.2 (Euler totient function). *Let n be an integer. The Euler totient function $\phi(n)$ measures the cardinality of the group $\mathbb{Z}_n^*$.*

Fact B.1. *If $n = pq$ is an RSA modulus, then $\phi(n) = (p-1)(q-1)$.*

Notation: We say that an RSA modulus $n = pq$ is chosen uniformly at random with security k if p and q are random k-bit primes. Let `RSAmodulus` denote the algorithm that, on input 1^k, outputs a random RSA modulus with security k. Let $\texttt{RSApk}(1^k)$ denote the algorithm that, on input 1^k, chooses two random k-bit primes, p and q, and outputs $n = pq$ and a random $e \in \mathbb{Z}_n^*$ such that $\gcd(e, \phi(n)) = 1$. Such a pair (n, e) is also called *RSA public key*, where the corresponding *secret key* is the value $d \in \mathbb{Z}_{\phi(n)}$ such that $ed \equiv 1 \bmod \phi(n)$.

Assumption B.1 (RSA Assumption). *The RSA assumption is that given an RSA public key (n, e), and a random element $u \in \mathbb{Z}_n^*$, it is hard to compute the value v such that $v^e \equiv u \bmod n$. More formally, we assume that for all polynomial-time circuit families $\{\mathcal{A}_k\}$, there exists a negligible function $\nu(k)$ such that*

$$\Pr[(n,e) \leftarrow \texttt{RSApk}(1^k); u \leftarrow \mathbb{Z}_n^*; v \leftarrow \mathcal{A}_k(n,e,u) \ : \ v^e \equiv u \bmod n] = \nu(k)$$

The strong RSA assumption was introduced by Barić and Pfitzmann [2] and Fujisaki and Okamoto [18].

Assumption B.2 (Strong RSA Assumption). *The strong RSA assumption is that it is hard, on input an RSA modulus n and an element $u \in \mathbb{Z}_n^*$, to compute values $e > 1$ and v such that $v^e \equiv u \bmod n$. More formally, we assume that for all polynomial-time circuit families $\{\mathcal{A}_k\}$, there exists a negligible function $\nu(k)$ such that*

$$\Pr[n \leftarrow \texttt{RSAmodulus}(1^k); u \leftarrow \mathbb{Z}_n^*; (v,e) \leftarrow \mathcal{A}_k(n,u) \ : e > 1 \ \wedge \ v^e \equiv u \bmod n] = \nu(k)$$

The tuple (n, u) generated as above, is called a *general instance* of the *flexible* RSA problem.

Notation: By $QR_n \subseteq \mathbb{Z}_n^*$ we will denote the set of quadratic residues modulo n, i.e., elements $a \in \mathbb{Z}_n^*$ such that $\exists b \in \mathbb{Z}_n^*$ such that $b^2 \equiv a \bmod n$.

If (n, u) is a general instance of the flexible RSA problem, and $u \in QR_n$, then (n, u) is a *quadratic* instance of the flexible RSA problem.

Note that since one quarter of all the elements of $\mathbb{Z}_n^*$ are quadratic residues, the strong RSA assumption implies that it is hard to solve quadratic instances of the flexible RSA problem. In the sequel, by an *instance* of the flexible RSA problem, we will mean a *quadratic*, rather than general instance.

Corollary B.1. *Under the strong RSA assumption, it is hard, on input a flexible RSA instance (n, u), to compute integers $e > 1$ and v such that $v^e \equiv u \bmod n$.*

Definition B.3 (Safe primes). *A prime number p is called* safe *if $p = 2p' + 1$, such that p' is also a prime number. (The corresponding number p' is known as a* Sophie Germain *prime.)*

Definition B.4 (Special RSA modulus). *An RSA modulus $n = pq$ is* special *if $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes.*

Notation: We say that a special RSA modulus $n = pq$ was chosen at random with security k if p and q are random k-bit safe primes.

Theorem B.2 (Chinese Remainder Theorem). *If $n = pq$, and $\gcd(p, q) = 1$, then $\Phi : \mathbb{Z}_n \mapsto \mathbb{Z}_p \times \mathbb{Z}_q$, $\Phi(x) = (x \bmod p, x \bmod q)$ is an efficiently computable and invertible ring isomorphism.*

Lemma B.3. *If $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$ is a special RSA modulus, then QR_n is a cyclic group under multiplication, of size $p'q'$, where all but $p' + q'$ of the elements are generators.*

Corollary B.4. *If (n, u) is an instance of the flexible RSA problem, and n is a special RSA modulus, we may assume that u is a generator of QR_n.*

The following lemma originates from the analysis of the primality test due to Miller [24] and Rabin [26].

Lemma B.5. *Let a composite integer n be given. Given any value x such that $\phi(n) \mid x$, one can find a non-trivial divisor of n in probabilistic polynomial time.*

Corollary B.6. *Let n be an integer. Let e such that $\gcd(e, \phi(n)) = 1$ be given. Given any value x such that $\phi(n) \mid x$, one can efficiently compute v such that $v^e \equiv u \bmod n$.*

For special RSA moduli, it is also true that given $x > 4$ such that $x \mid \phi(n)$, one can factor n. The following lemma and proof are straightforward:

Lemma B.7. *Let $n = pq$ be a special RSA modulus, i.e., $p = 2p' + 1$ and $q = 2q' + 1$, and p' and q' are primes. Suppose a value $x > 4$ is given such that $x \mid \phi(n) = 4p'q'$. Then it is possible to efficiently factor n.*

Proof. Suppose $x = 2^I p'$, for $I \in \{0, 1, 2\}$. Then factoring n is immediate. The more difficult case is $x = 2^I p'q'$. This immediately gives us $\phi(n)$. By Lemma B.5, we are done. □

Corollary B.8. *Given a special RSA modulus n, and an integer x such that $\gcd(\phi(n), x) > 4$, one can efficiently factor n.*

The following lemma is due to Shamir [30]:

Lemma B.9. *Let an integer n be given. Suppose that we are given the values $u, v \in \mathbb{Z}_n^*$ and $x, y \in \mathbb{Z}$, $\gcd(x, y) = 1$ such that $v^x \equiv u^y \bmod n$. Then there is an efficient procedure to compute the value z such that $z^x = u \bmod n$.*

The following lemma is a generalization of Lemma B.9, in that we are not restricted to the case where $\gcd(x, y) = 1$. However, note that this generalization applies only for special RSA moduli.

Lemma B.10. *Let a special RSA modulus* $n = pq$*,* $p = 2p'+1$*,* $q = 2q'+1$*, be given. Suppose we are given the values* $u, v \in QR_n$ *and* $x, y \in Z$*,* $\gcd(x, y) < x$ *such that* $v^x \equiv u^y \bmod n$*. Values* z*,* $w > 1$ *such that* $z^w \equiv u \bmod n$ *can be computed efficiently.*

Proof. Let $c = \gcd(x, y)$. If $\gcd(4c, \phi(n)) > 4$, then by Corollary B.8, we factor n. Otherwise, it must be the case that $\gcd(c, p'q') = 1$. Therefore, there exists a value $d \in \mathbb{Z}^*_{p'q'}$ such that $cd \equiv 1 \bmod p'q'$.

By the extended GCD algorithm, find integers a and b such that $ax + by = c$. Note that

$$v^{x/c} \equiv (v^x)^d \equiv (u^y)^d \equiv u^{y/c} \bmod n$$

Then, by Lemma B.9, we find an element z such that $z^{x/c} = u$ and we obtain $e = x/c$ and z. $\square$

Common inputs: Public key (n, a, b, c) of the signature scheme with parameters ℓ_m, ℓ_s, ℓ_e

Public key (g, h) of a commitment scheme modulo n

Commitment public key (n_C, g_C, h_C), commitment C_x

User's input: The values (x, r_x) such that $C_x = g_C^x h_C^{r_x} \bmod n_C$

and x is of length ℓ_m and r_x is of length ℓ_n.

Values (s, e, v), a valid signature on message x.

$U \longleftrightarrow V$ Choose, uniformly at random, values w, r_s, r_e, r_w, r_z and r of length ℓ_n. Compute the following quantities: $C_s = g^s h^{r_s} \bmod n$, $C_e = g^e h^{r_e} \bmod n$, $C_v = v g^w \bmod n$, $C_w = g^w h^{r_w}$, $z = ew$, $C_z = g^z h^{r_z}$. $C = (C_v)^e h^r \bmod n$.

It is important to note that $C = (C_v)^e h^r = v^e g^{we} h^r = (a^x b^s c) g^z h^r \bmod n$.

Send the values $(C_s, C_e, C_v, C_w, C_z, C)$ to the verifier and carry out the following zero-knowledge proofs of knowledge:

1. Show that C is a commitment in bases (C_v, h) to the value committed to in the commitment C_e:

$$PK\{(\epsilon, \rho, \rho_e) : C = C_v^\epsilon h^\rho \ \wedge\ C_e = g^\epsilon h^{\rho_e}\}$$

2. Show that C/c is also a commitment in bases $((a, b, g), h)$, to the values committed to by commitments C_x,C_s,C_z, and that the power of h here is the same as in C:

$$\begin{aligned} PK\{(\xi, \sigma, \zeta, \epsilon, \rho_x, \rho_s, \rho_z, \rho) \quad &: \quad C/c = a^\xi b^\sigma g^\zeta h^\rho \ \wedge\ C_x = g^\xi h^{\rho_x} \ \wedge \\ &\quad\ C_s = g^\sigma h^{\rho_s} \ \wedge\ C_z = g^\zeta h^{\rho_z} \ \wedge\ C = (C_v)^\epsilon h^\rho\} \end{aligned}$$

3. Show that C_z is a commitment to the product of the values committed to by C_w and C_e:

$$\begin{aligned} &PK\{(\zeta, \omega, \epsilon, \rho_z, \rho_w, \rho_e, \rho') \quad : \quad C_z = g^\zeta h^{\rho_z} \ \wedge\ C_w = g^\omega h^{\rho_w} \ \wedge \\ &C_e = g^\epsilon h^{\rho_e} \ \wedge\ C_z = (C_w)^\epsilon h^{\rho'}\} \end{aligned}$$

4. Show that C_x is a commitment to an integer of length ℓ_m, C_s is a commitment to an integer of length ℓ_s, and C_e is a commitment to an integer in the interval $(2^{\ell_e - 1}, 2^{\ell_e})$.

Figure 2: Proof of Knowledge of a Signature