Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes

Jan Camenisch[1⋆] and Markus Michels[2⋆⋆]

[1] BRICS
Department of Computer Science, University of Aarhus,
Ny Munkegade, DK – 8000 Århus C, Denmark
camenisch@daimi.au.dk
[2] Entrust Technologies Europe, r3 security engineering ag,
Glatt Tower, CH – 8301 Glattzentrum, Switzerland
Markus.Michels@entrust.com

Abstract. We present the first efficient statistical zero-knowledge protocols to prove statements such as:

- A committed number is a prime.
- A committed (or revealed) number is the product of two safe primes, i.e., primes p and q such that $(p-1)/2$ and $(q-1)/2$ are prime.
- A given integer has large multiplicative order modulo a composite number that consists of two safe prime factors.

The main building blocks of our protocols are statistical zero-knowledge proofs of knowledge that are of independent interest. We show how to prove the correct computation of a modular addition, a modular multiplication, and a modular exponentiation, where all values including the modulus are committed to but *not* publicly known. Apart from the validity of the equations, no other information about the modulus (e.g., a generator whose order equals the modulus) or any other operand is exposed. Our techniques can be generalized to prove that any multivariate modular polynomial equation is satisfied, where only commitments to the variables of the polynomial and to the modulus need to be known. This improves previous results, where the modulus is publicly known. We show how these building blocks allow to prove statements such as those listed earlier.

1 Introduction

The problem of proving that a number n is the product of two primes p and q of special form arises in many recently proposed cryptographic schemes (e.g., [7,8,20,21]) whose security is based on both the infeasibility of computing discrete logarithms and of computing roots in groups of unknown order. Such schemes typically involve a designated entity that knows the group's order and hence

⋆ BRICS - Basic Research in Computer Science, Center of the Danish National Research Foundation.

⋆⋆ Part of this work was done while this author was with Ubilab, UBS, Switzerland.

J. Stern (Ed.): EUROCRYPT'99, LNCS 1592, pp. 107–122, 1999.

is able to compute roots. Although the other involved entities must not learn the group's order, nevertheless, they want to be assured that it is large and not smooth, i.e., that computing discrete logarithms is infeasible to the designated entity as well. An example of groups used in such schemes are subgroups of $\mathbb{Z}_n^*$. Here, it suffices that the designated entity proves n to be the product of two safe primes, i.e., primes p and q such that $(p-1)/2$ and $(q-1)/2$ are prime. More precisely, if n is the product of two safe primes p and q and $a^2 \not\equiv 1 \pmod{n}$ and $\gcd(a^2-1, n) = 1$ holds for some a (which the verifier can check easily), then a has multiplicative order $(p-1)(q-1)/4$ or $(p-1)(q-1)/2$ [21]. Another example are elliptic curves over $\mathbb{Z}_n$. In this case, n is required to be the product of two primes p and q such that $(p+1)/2$ and $(q+1)/2$ are prime [25]. Finally, standards such as X9.31 require the modulus to be the product of two primes p and q, where $(p-1)/2$, $(p+1)/2$, $(q-1)/2$, and $(q+1)/2$ have a large prime factor that is between 100 and 120 bit [39][1]. Previously, the only way known to prove such properties was applying inefficient general zero-knowledge proof techniques (e.g., [23,5,16]).

In this paper we describe an efficient protocol for proving that a committed integer is in fact the modular addition of two committed integer modulo another committed integer without revealing any other information whatsoever. Then, we provide similar protocols for modular multiplication, modular exponentiation, and, more general, for any multivariate polynomial equation. Previously known protocols allow only to prove that algebraic relations modulo a *publicly known* integer hold [4,9,16,18]. Furthermore, we present an efficient zero-knowledge argument of primality of a committed number and, as a consequence, a zero-knowledge argument that an RSA modulus n consists of two safe primes. The additional advantage of this method is that only a commitment to n but not n itself must be publicly known. If the number n is publicly known, however, more efficient protocols can be obtained by combining our techniques with known results which are described in the next paragraph.

A number of protocols for proving properties of composite numbers are found in literature. Van de Graaf and Peralta [37] provide an efficient proof that a given integer n is of the form $n = p^r q^s$, where r and s are odd, p and q are primes and $p \equiv q \equiv 3 \pmod{4}$. A protocol due to Boyar et al. [2] allows to prove that a number n is square-free, i.e., there is no prime p with $p|n$ such that $p^2|n$. Hence, if both properties are proved, it follows that n is the product of two primes p and q, where $p \equiv q \equiv 3 \pmod{4}$. This result was recently strengthened by Gennaro et al. [22] who present a proof system for showing that a number n (satisfying certain side-conditions) is the product of quasi-safe primes, i.e., primes p and

[1] However, it is unnecessary to explicitly add this requirement to the RSA key generation. For randomly chosen large primes, the probability that $(p-1)/2$, $(p+1)/2$, $(q-1)/2$, and $(q+1)/2$ have a large prime factor is overwhelming. This is sufficient protection against the Pollard $p-1$ and Williams $p+1$ factoring methods [32,38]. Moreover, an efficient proof that an arbitrarily generated RSA modulus is not weak without revealing its factors seems to be hard to obtain as various conditions have to be checked (e.g., see [1]).

q for which $(p-1)/2$ and $(q-1)/2$ is a prime *power*. However, their protocol can not guarantee that $(p-1)/2$ and $(q-1)/2$ are indeed primes which is what we are aiming for. Finally, Chan et al. [11] and Mao [29] provide protocols for showing that a committed number consists of two large factors, and, recently, Liskov & Silverman describe a proof that a number is a product of two nearly equal primes [28].

2 Tools

In the following we assume a group $G = \langle g \rangle$ of large known order Q and a second generator h whose discrete logarithm to the base g is not known. We define the discrete logarithm of y to the base g to be any integer x such that $y = g^x$ holds, in particular discrete logarithms are allowed to be negative. Computing discrete logarithms is assumed to be infeasible.

2.1 Commitment Schemes

Our schemes use commitment schemes that allow to prove algebraic properties of the committed value. There are two kinds of commitment schemes. The first kind hides the committed value information theoretically from the verifier (unconditionally hiding) but is only conditionally binding, i.e., a computationally unbounded prover can change his mind. The second kind is only computationally hiding but unconditionally binding. Depending on the kind of the commitment scheme employed, our schemes will be statistical zero-knowledge arguments (proofs of knowledge) or computational zero-knowledge proof systems. Cramer and Damgård [16] describe a class of commitment schemes allowing to prove algebraic properties of the committed value. It includes RSA-based as well as discrete-logarithm-based schemes of both kinds. For easier description of our protocols, we will use a particular commitment scheme which is due to Pedersen [31]: A value $a \in \mathbb{Z}_Q$ is committed to by $c_a := g^a h^r$, where r is randomly chosen from $\mathbb{Z}_Q$. This scheme is unconditionally hiding and computationally binding, i.e., a prover able to compute $\log_g h$ can change his mind. Therefore our protocol will be statistical zero-knowledge proofs of knowledge (or arguments). However, our protocols can easily be adapted to work for all the commitment scheme exposed in [16].

2.2 Various Proof-Protocols Found in Literature

We review various zero-knowledge protocols for proving knowledge of and about discrete logarithms and introduce our notation for such protocols.

Proving the knowledge of a discrete logarithm x of a group element y to a base g [13,35]. The prover chooses a random $r \in_R \mathbb{Z}_Q$ and computes $t := g^r$ and sends t to the verifier. The verifier picks a random challenge $c \in_R \{0,1\}^k$ and sends it to the prover. The prover computes $s := r - cx \pmod{Q}$ and sends

s to the verifier. The verifier accepts, if $g^s y^c = t$ holds. This protocol is an *honest-verifier zero-knowledge proof of knowledge* if $k = \Theta(\text{poly}(\log Q))$ and a *zero-knowledge proof of knowledge* if $k = \mathcal{O}(\log\log(Q))$ and when serially repeated $\Theta(\text{poly}(\log Q))$ times. This holds for all other protocols described in this section (when not mentioned otherwise). Adopting the notation in [8], we denote this protocol by $PK\{(\alpha) : y = g^\alpha\}$, where PK stands for "proof of knowledge".

Proving the knowledge of a representation of an element y to the bases $g_1, \ldots, g_l$ [3,12], i.e., proving the knowledge of integers $x_1, \ldots, x_l$ such that $y = \prod_{i=1}^{l} g_i^{x_i}$. This protocol is an extension of the previous one with respect to multiple bases. The prover chooses random integers $r_1, \ldots, r_l \in_R \mathbb{Z}_Q$, computes $t := \prod_{i=1}^{l} g_i^{r_i}$, and sends the verifier t. The verifier returns her a randomly picked challenge $c \in_R \{0,1\}^k$. The prover computes $s_i := r_i - cx_i \pmod{Q}$ for $i = 1, \ldots, l$ and sends the verifier all s_i's, who accepts, if $t = y^c \prod_{i=1}^{l} g_i^{s_i}$ holds. This protocol is denoted by $PK\{(\alpha_1, \ldots, \alpha_l) : y = \prod_{i=1}^{l} g_i^{\alpha_i}\}$.

Proving the equality of the discrete logarithms of elements y_1 and y_2 to the bases g and h, respectively [14]. Let $y_1 = g^x$ and $y_2 = h^x$. The prover chooses a random $r \in \mathbb{Z}_Q^*$, computes $t_1 := g^r, t_2 := h^r$, and sends t_1, t_2 to the verifier. The verifier picks a random challenge $c \in \{0,1\}^k$ and sends it to the prover. The prover computes $s := r - cx \pmod{Q}$ and sends s to the verifier. The verifier accepts, if $g^s y_1^c = t_1$ and $h^s y_2^c = t_2$ holds. This protocol is denoted by $PK\{(\alpha) : y_1 = g^\alpha \wedge y_2 = h^\alpha\}$. Note that this method allows also to prove that one discrete log is the square of another one (modulo the group order), e.g., $PK\{(\alpha) : y_1 = g^\alpha \wedge y_2 = y_1^\alpha\}$.

Proving the knowledge of (at least) one out of the discrete logarithms of the elements y_1 and y_2 to the base g (proof of OR) [17,34]. W.l.o.g., we assume that the prover knows $x = \log_g y_1$. Then $r_1, s_2 \in_R \mathbb{Z}_Q^*$, $c_2 \in_R \{0,1\}^k$ and computes $t_1 := g^{r_1}, t_2 := g^{s_2} y_2^{c_2}$ and sends t_1 and t_2 to the verifier. The verifier picks a random challenge $c \in \{0,1\}^k$ and sends it to the prover. The prover computes $c_1 := c \oplus c_2$ and $s_1 := r_1 - c_1 x \pmod{Q}$ (where $\oplus$ denotes the bit-wise XOR operation) and sends s_1, s_2, c_1, and c_2 to the verifier. The verifier accepts, if $c_1 \oplus c_2 = c$ and $t_i = g^{s_i} y_i^{c_i}$ holds for $i \in \{1,2\}$. This protocol is denoted by $PK\{(\alpha, \beta) : y_1 = g^\alpha \ \vee \ y_2 = g^\beta\}$. This approach can be extended to an efficient system for proving arbitrary monotone statements built with $\wedge$'s and $\vee$'s [17,34].

Proving the knowledge of a discrete logarithm that lies in a given range, that is, $2^{\ell_1} - 2^{\ell_2} < \log_g y < 2^{\ell_1} + 2^{\ell_2}$, for some parameters ℓ_1 and ℓ_2. (The parameter 2^{ℓ_1} acts as an offset and can also chosen to be zero.) In principle, this statement can be proved by first committing to every bit of $x = \log_g y$ and then showing that the committed values are either a 0 or a 1 and constitute the binary representation of x. This method is linear in the number of bits of x. A more efficient but only statistical zero-knowledge protocol can be obtained from the basic protocol proving the knowledge of $\log_g y$ by restricting the verifier to binary challenges and by requiring the prover's response s to satisfy $2^{\ell_1} - 2^{\epsilon\ell_2+1} < s < 2^{\ell_1} + 2^{\epsilon\ell_2+1}$, where $\epsilon > 1$ is a security parameter. Now, when

considering how the knowledge extractor can compute an $x = \log_g y$ from two accepting protocol views with the same first message, it can be concluded that the prover must know an $x = \log_g y$ such that $2^{\ell_1} - 2^{\epsilon\ell_2+2} < x < 2^{\ell_1} + 2^{\epsilon\ell_2+2}$ holds [11]. We denote this protocol by

$$PK\{(\alpha) : y = g^\alpha \wedge 2^{\ell_1} - 2^{\ddot{\ell}_2} < \alpha < 2^{\ell_1} + 2^{\ddot{\ell}_2}\},$$

where $\ddot{\ell}_2$ denotes $\epsilon\ell_2+2$ (we will stick to that notation for the rest of the paper). For more details on this protocol we refer to [6,11]. Finally, the restriction to binary challenges can be dropped if the order of the group is not known to the prover (e.g., if a subgroup of an RSA-ring is used) and when believing in the non-standard strong RSA-assumption[2] [18,19]. Although we describe our protocols in the following in the setting where the group's order is known to the prover, all protocols can easily be adapted to the case where the prover does not know the group's order using the techniques from [18,19].

All described protocols can be combined in natural ways. First of all, one can use multiple bases instead of a single one in any of the preceding protocols. Then, executing any number of instances of these protocols in parallel and choosing the same challenges for all of them in each round corresponds to the $\wedge$-composition of the statements the single protocols prove. Using this approach, it is even possible to compose instances according to any monotone formula [17,34]. In the following we will use of such compositions without having explained the technical details involved for which we refer to [4,9,10,17,34].

3 Secret Computations with a Secret Modulus

The goal of this section is to provide an efficient protocol to prove that $a^b \equiv d \pmod{n}$ holds for some committed integers without revealing the verifier any further information (i.e., the protocol is zero-knowledge). A step towards this goal are protocols to prove that a committed integer is the addition or the multiplication of two other committed integers modulo a third committed integer n.

The algebraic setting is as follows. Let ℓ be an integer such that $-2^\ell < a, b, d, n < 2^\ell$ holds and $\epsilon > 1$ be security parameter (cf. Section 2). Furthermore, we assume that a group G of order $Q > 2^{2\epsilon\ell+5}$ $(= 2^{2\ddot{\ell}+1})$ and two generators g and h are available such that $\log_g h$ is not known. This group could for instance be chosen by the prover in which case she would have to prove that she has chosen it correctly. Finally, let the prover's commitments to a, b, d, and n be $c_a := g^a h^{r_1}$, $c_b := g^b h^{r_2}$, $c_d := g^d h^{r_3}$, and $c_n := g^n h^{r_4}$, where r_1, r_2, r_3, and r_4 are randomly chosen elements of $\mathbb{Z}_Q$.

[2] The strong RSA assumption states that there exists a probabilistic polynomial-time algorithm G that on input $1^{|n|}$ outputs an RSA-modulus n and an element $z \in \mathbb{Z}_n^*$ such that it is infeasible to find integers $e \notin \{-1, 1\}$ and u such that $z \equiv u^e \pmod{n}$.

3.1 Secret Modular Addition and Multiplication

We assume that the verifier already obtained the commitments c_a, c_b, c_d, and c_n. Then the prover can convince the verifier that $a + b \equiv d \pmod{n}$ holds by sequentially running the protocol denoted[3] by

$$S_+ := PK\big\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\eta,\vartheta,\pi,\lambda):\ c_a = g^{\alpha}h^{\beta} \ \wedge\ (-2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}}) \wedge \\ c_b = g^{\gamma}h^{\delta} \ \wedge\ (-2^{\ddot{\ell}} < \gamma < 2^{\ddot{\ell}}) \ \wedge\ c_d = g^{\varepsilon}h^{\zeta} \ \wedge\ (-2^{\ddot{\ell}} < \varepsilon < 2^{\ddot{\ell}}) \wedge \\ c_n = g^{\eta}h^{\vartheta} \ \wedge\ (-2^{\ddot{\ell}} < \eta < 2^{\ddot{\ell}}) \ \wedge\ \frac{c_d}{c_a c_b} = c_n^{\pi}h^{\lambda} \ \wedge\ (-2^{\ddot{\ell}} < \pi < 2^{\ddot{\ell}})\big\}$$

k times. Alternatively, she can convince the verifier that $ab \equiv d \pmod{n}$ holds by running the protocol

$$S_* := PK\big\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\eta,\vartheta,\varrho,\varsigma):\ c_a = g^{\alpha}h^{\beta} \ \wedge\ (-2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}}) \wedge \\ c_b = g^{\gamma}h^{\delta} \ \wedge\ (-2^{\ddot{\ell}} < \gamma < 2^{\ddot{\ell}}) \ \wedge\ c_d = g^{\varepsilon}h^{\zeta} \ \wedge\ (-2^{\ddot{\ell}} < \varepsilon < 2^{\ddot{\ell}}) \wedge \\ c_n = g^{\eta}h^{\vartheta} \ \wedge\ (-2^{\ddot{\ell}} < \eta < 2^{\ddot{\ell}}) \ \wedge\ c_d = c_b^{\alpha}c_n^{\varrho}h^{\varsigma} \ \wedge\ (-2^{\ddot{\ell}} < \varrho < 2^{\ddot{\ell}})\big\}$$

k times with him.

Remark. In some applications the prover might be required to show that n has some minimal size. This can by showing that η lies in the range $2^{\ell_1} - 2^{\ddot{\ell}_2} < \eta < 2^{\ell_1} + 2^{\ddot{\ell}_2}$ instead of $-2^{\ddot{\ell}} < \eta < 2^{\ddot{\ell}}$ for some appropriate values of ℓ_1 and ℓ_2 (cf. Section 2.2).

Theorem 1. *Let a, b, d, and n be integers that are committed to by the prover as described above and assume computing discrete logarithms in G is infeasible. Then the protocol S_+ is a statistical zero-knowledge argument that $a + b \equiv d \pmod{n}$ holds. Furthermore, the protocol S_* is a statistical zero-knowledge argument that $ab \equiv d \pmod{n}$ holds. The soundness error probability for both protocols is 2^{-k}.*

Proof. The statistical zero-knowledge claims follows from the statistical zero-knowledgeness of the building blocks.

Let us argue why the modular relations among the committed integers hold. First, we consider what the clauses prove that S_+ and S_* have in common. Running the prover with either protocol (and using standard techniques), the knowledge extractor can compute integers $\hat{a}$, $\hat{b}$, $\hat{d}$, $\hat{n}$, $\hat{r}_1$, $\hat{r}_2$, $\hat{r}_3$, and $\hat{r}_4$ such that $c_a = g^{\hat{a}}h^{\hat{r}_1}$, $c_b = g^{\hat{b}}h^{\hat{r}_2}$, $c_d = g^{\hat{b}}h^{\hat{r}_3}$, and $c_n = g^{\hat{n}}h^{\hat{r}_4}$ holds. Moreover, $-2^{\ddot{\ell}} < \hat{a} < 2^{\ddot{\ell}}$, $-2^{\ddot{\ell}} < \hat{b} < 2^{\ddot{\ell}}$, $-2^{\ddot{\ell}} < \hat{d} < 2^{\ddot{\ell}}$, and $-2^{\ddot{\ell}} < \hat{n} < 2^{\ddot{\ell}}$ holds for these integers.

When running the prover with S_+, the knowledge extractor can further compute integers $\hat{r}_5 \in \mathbb{Z}_Q$ and $\hat{u}$ with $-2^{\ddot{\ell}} < \hat{u} < 2^{\ddot{\ell}}$ such that $c_d/(c_a c_b) = c_n^{\hat{u}}h^{\hat{r}_5}$ holds. Therefore, we have $g^{\hat{d}-\hat{a}-\hat{b}}h^{\hat{r}_3-\hat{r}_1-\hat{r}_2} = g^{\hat{n}\hat{u}}h^{\hat{u}\hat{r}_4+\hat{r}_5}$ and hence, provided

[3] Recall that $\ddot{\ell}$ denotes $\epsilon\ell + 2$.

$\log_g h$ is not known, we must have $\hat{d} \equiv \hat{a} + \hat{b} + \hat{u}\hat{n} \pmod{Q}$. Thus we have $\hat{d} = \hat{a} + \hat{b} + \hat{u}\hat{n} + \bar{w}Q$ for some integer $\bar{w}$. Since $2^{2\ddot{\ell}+1} < Q$ and due to the constraints on $\hat{a}$, $\hat{b}$, $\hat{d}$, $\hat{n}$, and $\hat{u}$, we can conclude that the integer $\bar{w}$ must be 0 and so $\hat{d} \equiv \hat{a} + \hat{b} \pmod{\hat{n}}$ must hold.

Now consider the case when running the prover with S_*. In this case the knowledge-extractor can additionally compute integers $\hat{r}_6 \in \mathbb{Z}_Q$ and $\hat{v}$ with $-2^{\ddot{\ell}} < \hat{v} < 2^{\ddot{\ell}}$ such that $c_d = c_b^{\hat{a}} c_n^{\hat{v}} h^{\hat{r}_6}$ and thus $g^{\hat{d}} h^{\hat{r}_3} = g^{\hat{a}\hat{b}+\hat{v}\hat{n}} h^{\hat{a}\hat{r}_2+\hat{v}\hat{r}_4+\hat{r}_6}$ holds. Again, assuming that $\log_g h$ is not known, we have $\hat{d} \equiv \hat{a}\hat{b} + \hat{v}\hat{n} \pmod{Q}$. As before, due to $2^{2\ddot{\ell}+1} < Q$ and the constraints on $\hat{a}$, $\hat{b}$, $\hat{d}$, $\hat{n}$, and $\hat{v}$ we can conclude that $\hat{d} \equiv \hat{a}\hat{b} \pmod{\hat{n}}$ must hold for the committed values. □

3.2 Secret Modular Exponentiation

We now extend the ideas from the previous paragraph to a method for proving that $a^b \equiv d \pmod{n}$ holds. Using the same approach as above, i.e., having the prover to provide a commitment to an integer $\tilde{a}$ that equals a^b (in $\mathbb{Z}$) and proving this, would required that G has order about $2^{b\ell}$ and thus such a protocol would become rather inefficient. A more efficient protocol is obtained by constructing $a^b \pmod{n}$ step by step according to the square & multiply algorithm[4], committing to all intermediary results, and then prove that everything is consistent. This protocol is exposed in the following. We assume that an upper-bound $\ell_b \leq \ell$ on the length of b is publicly known.

1. Apart from her commitments c_a, c_b, c_d, and c_n to a, $b = \sum_{i=0}^{\ell_b-1} b_i 2^i$, d, and n, the prover must commit to all the bits of b: let $c_{b_i} := g^{b_i} h^{\tilde{r}_i}$ with $\tilde{r}_i \in_R \mathbb{Z}_Q$ for $i \in \{0, \ldots, \ell_b - 1\}$. Furthermore she needs to provide commitments to the intermediary results of the square & multiply algorithm: let $c_{v_i} := g^{(a^{2^i} \pmod{n})} h^{\hat{r}_i}$, $(i = 1, \ldots, \ell_b - 1)$, be her commitments to the powers of a, i.e., $a^{2^i} \pmod{n}$, where $\hat{r}_i \in_R \mathbb{Z}_Q$, and let $c_{u_i} := g^{u_i} h^{\bar{r}_i}$, $(i = 0, \ldots, \ell_b - 2)$, where $u_i := u_{i-1}(a^{2^i})^{b_i} \pmod{n}$, $(i = 1, \ldots, \ell_b - 2)$, $u_0 = a^{b_0} \pmod{n}$, and $\bar{r}_i \in_R \mathbb{Z}_Q$. The prover sends the verifier all these commitments.
2. To prove that $a^b \equiv d \pmod{n}$ holds, they carry out the following protocol k times.

$$S_\uparrow := PK\Big\{ \big(\alpha, \beta, \xi, \chi, \gamma, \delta, \varepsilon, \zeta, \eta, (\lambda_i, \mu_i, \nu_i, \xi_i, \varsigma_i, \tau_i, \vartheta_i, \varphi_i, \psi_i)_{i=1}^{\ell_b-1}, (\pi_i, \varrho_i)_{i=1}^{\ell_b-2}\big) :$$

$$c_a = g^{\alpha} h^{\beta} \ \wedge \ -2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}} \ \wedge \tag{1}$$

$$c_d = g^{\gamma} h^{\delta} \ \wedge \ -2^{\ddot{\ell}} < \gamma < 2^{\ddot{\ell}} \ \wedge \tag{2}$$

$$c_n = g^{\varepsilon} h^{\zeta} \ \wedge \ -2^{\ddot{\ell}} < \varepsilon < 2^{\ddot{\ell}} \ \wedge \tag{3}$$

$$\Big(\prod_{i=0}^{\ell_b-1} c_{b_i}^{2^i}\Big)/c_b = h^{\eta} \ \wedge \tag{4}$$

[4] In practice a more enhanced exponentiation algorithm might be used (see, e.g., [15]), but one should keep in mind that it must not leak additional information about the exponent.

$$c_{v_1} = g^{\lambda_1} h^{\mu_1} \ \wedge \ \ldots \ \wedge \ c_{v_{\ell_b-1}} = g^{\lambda_{\ell_b-1}} h^{\mu_{\ell_b-1}} \ \wedge \tag{5}$$

$$c_{v_1} = c_a^{\alpha} c_n^{\nu_1} h^{\xi_1} \ \wedge \ c_{v_2} = c_{v_1}^{\lambda_1} c_n^{\nu_2} h^{\xi_2} \ \wedge \ \ldots \ \wedge \ c_{v_{\ell_b-1}} = c_{v_{\ell_b-2}}^{\lambda_{\ell_b-2}} c_n^{\nu_{\ell_b-1}} h^{\xi_{\ell_b-1}} \ \wedge \tag{6}$$

$$-2^{\ddot{\ell}} < \lambda_1 < 2^{\ddot{\ell}} \ \wedge \ldots \ \wedge \ -2^{\ddot{\ell}} < \lambda_{\ell_b-1} < 2^{\ddot{\ell}} \ \wedge \tag{7}$$

$$-2^{\ddot{\ell}} < \nu_1 < 2^{\ddot{\ell}} \ \wedge \ldots \ \wedge \ -2^{\ddot{\ell}} < \nu_{\ell_b-1} < 2^{\ddot{\ell}} \ \wedge \tag{8}$$

$$c_{u_1} = g^{\pi_1} h^{\varrho_1} \ \wedge \ \ldots \ \wedge \ c_{u_{\ell_b-2}} = g^{\pi_{\ell_b-2}} h^{\varrho_{\ell_b-2}} \ \wedge \tag{9}$$

$$-2^{\ddot{\ell}} < \pi_1 < 2^{\ddot{\ell}} \ \wedge \ldots \ \wedge \ -2^{\ddot{\ell}} < \pi_{\ell_b-2} < 2^{\ddot{\ell}} \ \wedge \tag{10}$$

$$\Big(\big(c_{b_0} = h^{\varsigma_0} \ \wedge \ c_{u_0}/g = h^{\tau_0} \big) \vee \big(c_{b_0}/g = h^{\vartheta_0} \ \wedge \ c_{u_0}/c_a = h^{\psi_0} \big) \Big) \wedge \tag{11}$$

$$\Big(\big(c_{b_1} = h^{\varsigma_1} \ \wedge \ c_{u_1}/c_{u_0} = h^{\tau_1} \big) \vee \tag{12}$$

$$\big(c_{b_1}/g = h^{\vartheta_1} \ \wedge \ c_{u_1} = c_{u_0}^{\lambda_1} c_n^{\varphi_1} h^{\psi_1} \ \wedge \ -2^{\ddot{\ell}} < \varphi_1 < 2^{\ddot{\ell}} \big) \Big) \wedge \ \ldots \ \wedge$$

$$\Big(\big(c_{b_{\ell_b-2}} = h^{\varsigma_{\ell_b-2}} \ \wedge \ c_{u_{\ell_b-2}}/c_{u_{\ell_b-3}} = h^{\tau_i} \big) \vee \tag{13}$$

$$\big(c_{b_{\ell_b-2}}/g = h^{\vartheta_{\ell_b-2}} \ \wedge \ c_{u_{\ell_b-2}} = c_{u_{\ell_b-3}}^{\lambda_{\ell_b-2}} c_n^{\varphi_{\ell_b-2}} h^{\psi_{\ell_b-2}} \ \wedge \ -2^{\ddot{\ell}} < \varphi_{\ell_b-2} < 2^{\ddot{\ell}} \big) \Big) \wedge$$

$$\Big(\big(c_{b_{\ell_b-1}} = h^{\varsigma_{\ell_b-1}} \ \wedge \ c_d/c_{u_{\ell_b-2}} = h^{\tau_i} \big) \vee \tag{14}$$

$$\big(c_{b_{\ell_b-1}}/g = h^{\vartheta_{\ell_b-1}} \ \wedge \ c_d = c_{u_{\ell_b-2}}^{\lambda_{\ell_b-1}} c_n^{\varphi_{\ell_b-1}} h^{\psi_{\ell_b-1}} \ \wedge \ -2^{\ddot{\ell}} < \varphi_{\ell_b-1} < 2^{\ddot{\ell}} \big) \Big) \Big\}$$

Let us now explain why this protocol proves that $a^b \equiv d \pmod{n}$ holds and consider the clauses of sub-protocol $S_\uparrow$. What the Clauses 1–3 prove should be clear. The Clause 4 shows that the c_{b_i}'s indeed commit to the bits of the integer committed to in c_b (that these are indeed bits is shown in the Clauses 11–14). From this it can further be concluded that c_b commits to a value smaller than 2^{ℓ_b}. The Clauses 5–8 prove that the c_{v_i}'s indeed contain $a^{2^i} \pmod{n}$ (cf. Section 3.1). Finally, the Clauses 9–14 show that c_{u_i}'s commit to the intermediary results of the square & multiply algorithm and that c_d commits to the result: The Clauses 9 and 10 show that the c_{u_i}'s commit to integers that lie in $\{-2^{\ddot{\ell}}+1, \ldots, 2^{\ddot{\ell}}-1\}$ (for c_{u_0} this follows from Clause 11). Then, Clause 11 proves that either c_{b_0} commits to a 0 and c_{u_0} commits to a 1 or c_{b_0} commits to a 1 and c_{u_0} commits to the same integer as c_a. The Clauses 12 and 13 show that for each $i = 1, \ldots, \ell_b - 2$ either c_{b_i} commits to a 0 and c_{u_i} commits to same integer as $c_{u_{i-1}}$ or c_{b_i} commits to a 1 and c_{u_i} commits to the modular product of the value $c_{u_{i-1}}$ commits to and of $a^{2^i} \pmod{n}$ (which c_{v_i} commits to). Finally, Clause 14 proves (in a similar manner as the Clauses 12 and 13) that c_d commits to the result of the square & multiply algorithm and thus to $a^b \pmod{n}$.

Theorem 2. *Let c_a, c_b, c_d, and c_n be commitments to integers a, b, d, and n and let $c_{b_0}, \ldots, c_{b_{\ell-1}}, c_{v_1}, \ldots, c_{v_{\ell_b-1}}, c_{u_0}, \ldots, c_{u_{\ell_b-2}}$ be auxiliary commitments. Then, assuming computing discrete logarithms in G is infeasible, the protocol $S_\uparrow$ is a statistical zero-knowledge argument that the equation $a^b \equiv d \pmod{n}$ holds. The soundness error probability is 2^{-k}.*

Proof. The proof is straight forward from Theorem 1 and the explanations given above that $c_{b_0}, \ldots, c_{b_{\ell-1}}, c_{v_1}, \ldots, c_{v_{\ell_b-1}}, c_{u_0}, \ldots, c_{u_{\ell_b-2}}, S$ implement the square & multiply algorithm step by step. □

In the following, when denoting a protocol, we refer to the protocol $S_\uparrow$ by adding a clause like $\left(\alpha^\beta \equiv \gamma \pmod{\delta}\right)$ to the statement that is proven and assume that the prover sends the verifier all necessary commitments; e.g.,

$$PK\Big\{(\alpha,\beta,\gamma,\delta,\tilde{\alpha},\tilde{\beta},\tilde{\gamma},\tilde{\delta}) : c_a = g^\alpha h^{\tilde{\alpha}} \ \wedge \ c_b = g^\beta h^{\tilde{\beta}} \ \wedge \ c_d = g^\gamma h^{\tilde{\gamma}} \ \wedge \ c_n = g^\delta h^{\tilde{\delta}} \ \wedge \ \left(\alpha^\beta \equiv \gamma \pmod{\delta}\right)\Big\}.$$

3.3 Efficiency Analysis

We assume that G is chosen such that group elements can be represented with about $\log Q$ bits. For both S_+ and S_* the prover and the verifier both need to compute 5 multi-exponentiations per round. The communication per round is about $10 \log Q + 5\epsilon\ell$ bits in case of S_+ and S_*. In case of $S_\uparrow$, the prover and the verifier need to compute about $7\ell_b$ multi-exponentiations per round. Additionally, the prover needs to compute about $3\ell_b$ multi-exponentiations in advance of the protocol (these are the computations of the commitments to the intermediary results of the square & multiply algorithm). The communication cost per round is about $14\ell_b \log Q + 4\ell_b\epsilon\ell$ bits and an initial $3\ell_b$ group element which are the commitments to the intermediary results of the square & multiply algorithm.

3.4 Extension to a General Multivariate Polynomial

Let us outline how the correct computation of a general multivariate polynomial equation of form

$$f(x_1, \ldots, x_t, a_1, \ldots, a_l, b_{1,1}, \ldots, b_{l,t}, n) = \sum_{i=1}^{l} a_i \prod_{j=1}^{t} x_j^{b_{i,j}} \equiv 0 \pmod{n}$$

can be shown, where all integers $x_1, \ldots, x_t, a_1, \ldots, a_l, b_{1,1}, \ldots, b_{l,t}$, and n might only given as commitments: The prover commits to all the summands $s_1 := a_1 \prod_{j=1}^{t} x_j^{b_{1,j}} \pmod{n}, \ldots, s_l := a_l \prod_{j=1}^{t} x_j^{b_{l,j}} \pmod{n}$ and shows that the sum of these summands is indeed zero modulo n. Then, she commits to all the product terms $p_{1,1} := x_1^{b_{1,1}} \pmod{n}, \ldots, p_{t,l} := x_t^{b_{l,t}} \pmod{n}$ of the product and shows that $s_i \equiv a_i \prod_{j=1}^{t} p_{i,j} \pmod{n}$. Finally, she shows that $p_{i,j} \equiv x_j^{b_{i,j}} \pmod{n}$ (using the protocol $S_\uparrow$) and that for all i the same x_j is in $p_{i,j}$. This extends easily to several polynomial equations, where some variables appear in more than one equation.

4 A Proof that a Secret Number Is a Prime

In this section we describe how a prover and a verifier can carry out a primality test for an integer hidden in a commitment. Some primality tests reveal information about the structure of the prime and are hence not suited unless one is willing to expose this information. Examples of such tests are the Miller-Rabin test [30,33] or the one based on Pocklington's theorem. A test that does not reveal such information is due to Lehmann [27] and described in the next subsection.

4.1 Lehmann's Primality Test

Lehmann's test is variation of the Solovay-Strassen [36] primality test and based on the following theorem [26]:

Theorem 3. *An odd integer $n > 1$ is prime if and only if*

$$\forall a \in \mathbb{Z}_n^* : \; a^{(n-1)/2} \equiv \pm 1 \pmod{n} \text{ and } \exists a \in \mathbb{Z}_n^* : \; a^{(n-1)/2} \equiv -1 \pmod{n} .$$

This theorem suggest the following probabilistic primality test [27]:

- Choose k random bases $a_1, \ldots, a_k \in \mathbb{Z}_n^*$,
- check whether $a_i^{(n-1)/2} \equiv \pm 1 \pmod{n}$ holds for all i's, and
- check whether $a_i^{(n-1)/2} \equiv -1 \pmod{n}$ if true for at least one $i \in \{1, \ldots, k\}$.

The probability that a non-prime n passes this test is at most 2^{-k}, and that a prime n does not pass this test is at most 2^{-k} as well. Note that in case n and $(n-1)/2$ are both odd, the condition that $a_i^{(n-1)/2} \equiv -1 \pmod{n}$ holds for at least one i can be omitted. In this special case of Lehmann's test is equivalent to the Miller-Rabin test [33] and the failure probability is at most 4^{-k}.

4.2 Proving the Primality of a Committed Number

We now show how the prover and the verifier can apply Lehmann's primality test to a number committed to by the prover such that the verifier is convinced that the test was correctly done but does not learn any other information. The general idea is that the prover commits to t random bases a_i (of course, the verifier must be assured that the a_i's are chosen at random) and then proves that for these bases $a_i^{(n-1)/2} \equiv \pm 1 \pmod{n}$ holds. Furthermore, to conform with the second condition in Theorem 3, the prover must commit to a base, say $\tilde{a}$, such that $\tilde{a}^{(n-1)/2} \equiv -1 \pmod{n}$ holds.

Let ℓ be an integer such that $n < 2^\ell$ holds and let $\epsilon > 1$ be a security parameter. As in the previous section, a group G of prime order $Q > 2^{2\epsilon\ell+5}$ and two generators g and h are chosen, such that $\log_g h$ is not known. Let $c_n := g^n h^{r_n}$ with $r_n \in_R \mathbb{Z}_Q$ be the prover's commitment to the integer on which the primality test should be performed.

The following four steps constitute the protocol.

1. The prover picks random $\hat{a}_i \in_R \mathbb{Z}_n$ for $i = 1, \ldots, t$ and commits to them as $c_{\hat{a}_i} := g^{\hat{a}_i} h^{r_{\hat{a}_i}}$ with $r_{\hat{a}_i} \in_R \mathbb{Z}_Q$ for $i = 1, \ldots, t$. She sends $c_{\hat{a}_1}, \ldots, c_{\hat{a}_t}$ to the verifier.
2. The verifier picks random integers $-2^{\ell} < \check{a}_i < 2^{\ell}$ for $i = 1, \ldots, t$ and sends them to the prover.
3. The prover computes $a_i := \hat{a}_i + \check{a}_i \pmod{n}$, $c_{a_i} := g^{a_i} h^{r_{a_i}}$ with $r_{a_i} \in_R \mathbb{Z}_Q$, $d_i := a_i^{(n-1)/2} \pmod{n}$, and $c_{d_i} := g^{d_i} h^{r_{d_i}}$ with $r_{d_i} \in_R \mathbb{Z}_Q$ for all $i = 1, \ldots, t$. Moreover, the prover commits to $(n-1)/2$ by $c_b := g^{(n-1)/2} h^{r_b}$ with $r_b \in_R \mathbb{Z}_Q$. Then the prover searches a base $\tilde{a}$ such that $\tilde{a}^{(n-1)/2} \equiv -1 \pmod{n}$ holds and commits to $\tilde{a}$ by $c_{\tilde{a}} := g^{\tilde{a}} h^{r_{\tilde{a}}}$ with $r_{\tilde{a}} \in_R \mathbb{Z}_Q$.
4. The prover sends $c_b, c_{\tilde{a}}, c_{a_1}, \ldots, c_{a_t}, c_{d_1}, \ldots, c_{d_t}$ to the verifier and then they carry out the following (sub-)protocol k times.

$$S_p := PK\Big\{ \big(\alpha, \beta, \gamma, \nu, \xi, \varrho, \omega, (\delta_i, \varepsilon_i, \zeta_i, \eta_i, \vartheta_i, \pi_i, \varrho_i, \kappa_i, \mu_i, \psi_i)_{i=1}^{t}\big) :$$

$$c_b = g^{\alpha} h^{\beta} \ \wedge \ -2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}} \ \wedge \tag{15}$$

$$c_n = g^{\nu} h^{\xi} \ \wedge \ -2^{\ddot{\ell}} < \nu < 2^{\ddot{\ell}} \ \wedge \tag{16}$$

$$c_b^2 g / c_n = h^{\gamma} \ \wedge \tag{17}$$

$$c_{\tilde{a}} = g^{\varrho} h^{\omega} \ \wedge \ (\varrho^{\alpha} \equiv -1 \pmod{\nu}) \ \wedge \tag{18}$$

$$c_{\hat{a}_1} = g^{\delta_1} h^{\varepsilon_1} \ \wedge \ldots \wedge \ c_{\hat{a}_t} = g^{\delta_t} h^{\varepsilon_t} \ \wedge \tag{19}$$

$$c_{a_1} / g^{\check{a}_1} = g^{\delta_1} c_n^{\zeta_1} h^{\eta_1} \ \wedge \ldots \wedge \ c_{a_t} / g^{\check{a}_t} = g^{\delta_t} c_n^{\zeta_t} h^{\eta_t} \ \wedge \tag{20}$$

$$-2^{\ddot{\ell}} < \delta_1 < 2^{\ddot{\ell}} \ \wedge \ldots \wedge \ -2^{\ddot{\ell}} < \delta_t < 2^{\ddot{\ell}} \ \wedge \tag{21}$$

$$-2^{\ddot{\ell}} < \zeta_1 < 2^{\ddot{\ell}} \ \wedge \ldots \wedge \ -2^{\ddot{\ell}} < \zeta_t < 2^{\ddot{\ell}} \ \wedge \tag{22}$$

$$c_{a_1} = g^{\varrho_1} h^{\kappa_1} \ \wedge \ldots \wedge \ c_{a_t} = g^{\varrho_t} h^{\kappa_t} \ \wedge \tag{23}$$

$$\big(c_{d_1} / g = h^{\vartheta_1} \vee c_{d_1} g = h^{\vartheta_1}\big) \wedge \ldots \wedge \big(c_{d_t} / g = h^{\vartheta_t} \vee c_{d_t} g = h^{\vartheta_t}\big) \wedge \tag{24}$$

$$c_{d_1} = g^{\mu_1} h^{\psi_1} \ \wedge \ldots \wedge \ c_{d_t} = g^{\mu_t} h^{\psi_t} \ \wedge \tag{25}$$

$$(\varrho_1^{\alpha} \equiv \mu_1 \pmod{\nu}) \ \wedge \ldots \wedge \ (\varrho_t^{\alpha} \equiv \mu_t \pmod{\nu}) \Big\} \tag{26}$$

Let us analyze the protocol. In Step 1 and 2 of the protocol, the prover and the verifier together choose the random bases $a_1, \ldots, a_t$ for the primality test. Each base is the sum (modulo n) of the random integer the verifier chose and the one the prover chose. Hence, both parties are ensured that the bases are random, although the verifier does not get any information about the bases finally used in the primality test. That the bases are indeed chosen according to this procedure is shown in the Clauses 19–23 of the sub-protocol S_p, where the correct generation of the random values a_i, committed in c_{a_i}, is proved. The Clauses 16–17 prove that indeed $(n-1)/2$ is committed in c_b and the Clause 18 shows that there exists a base $\tilde{a}$ such that $\tilde{a}^{(n-1)/2} \equiv -1 \pmod{n}$. In the Clause 24 it is shown that the values committed in c_{d_i} are either equal to -1 or to 1. Finally, in Clause 26 (together with the Clauses 15, 16, 23, and 25) it is proved that $a_i^{(n-1)/2} \equiv d_i \pmod{n}$, i.e., $a_i^{(n-1)/2} \pmod{n} \in \{-1, 1\}$ and thus the conditions that n is a prime with error-probability 2^{-t} are met.

Note that all modular exponentiations in Clause 26 have the same b and n and hence the proofs for these parts can be optimized. In particular, this is the case for the Clauses 3, 4, and 11–14 in $S_{\uparrow}$.

Theorem 4. *Assume computing discrete logarithms in G is infeasible. Then, the above protocol is a statistical zero-knowledge argument that the integer committed to by c_n is a prime. The soundness error probability is at most $2^{-k}+2^{-t}$.*

Proof. The proof is straight forward from the Theorems 1, 2, and 3. □

In the sequel, we abbreviate the above protocol by adding a clause such as $\alpha \in \texttt{primes}(t)$ to the statement that is proven, where t denotes the number of bases used in the primality test.

Remark. If $(n-1)/2$ is odd and the prover is willing to reveal this, she can additionally prove that she knows χ and ψ such that $c_b/g = (g^2)^{\chi}h^{\psi}$ and $-2^{\ddot{\ell}} < \chi < 2^{\ddot{\ell}}$ holds and skip the Clause 18. This results in a statistical zero-knowledge proof that n of form $n = 2w+1$ is prime and w is odd with error-probability at most 2^{-2t}.

4.3 Efficiency Analysis

Assume that the commitment to the prime n is given. Altogether $t+1$ protocols that a modular exponentiation holds are carried out where the exponents are about $\log n$ bits. Thus, prover and verifier need to compute about $7t \log n$ multi-exponentiations per round each. Additionally, the prover needs to compute about $2t \log n$ multi-exponentiations for the commitments to the intermediary results of the square & multiply algorithm. (Note that the exponents in Clause 26 is the same in all relations and hence the commitments to its bits need to be computed only once.) The communication cost per round is about $14t \log n \log Q + 4t \log n \epsilon \ell$ bits and an initial $2t \log n$ group elements which are the commitments to the intermediary results of the square & multiply algorithm and the commitments to the bases of the primality test.

5 Proving that an RSA Modulus Consists of Two Safe Primes

We finally present protocols for proving that an RSA modulus consists of two safe primes. First, we restrict ourselves to the case where not the modulus but only a commitment to it is not known to the verifier. Later, we will discuss improvements for cases when the RSA modulus is known to the verifier.

5.1 A Protocol for a Secret RSA Modulus

Let 2^{ℓ} be an upper-bound on the length of the largest factor of the modulus and let $\epsilon > 1$ be a security parameter. Furthermore, a group G of prime order

$Q > 2^{2\epsilon\ell+5}$ and two generators g and h are chosen, such that $\log_g h$ is not known and computing discrete logarithms is infeasible. Let $c_n := g^n h^{r_n}$ be the prover's commitment to an integer n, where she has chosen $r_n \in_R \mathbb{Z}_Q$, and let p and q denote the two prime factors of n. The following is a protocol that allows her to convince the verifier that c_n commits to the product of two safe primes.

1. The prover computes the commitments $c_p := g^p h^{r_p}$, $c_{\tilde{p}} := g^{(p-1)/2} h^{r_{\tilde{p}}}$, $c_q := g^q h^{r_b}$, and $c_{\tilde{q}} := g^{(q-1)/2} h^{r_{\tilde{p}}}$ with $r_p, r_{\tilde{p}}, r_q, r_{\tilde{q}} \in_R \mathbb{Z}_Q$ and sends all these commitments to the verifier.
2. The two parties sequentially carry out the following protocol k times.

$$S_{51} := PK\{(\alpha, \beta, \gamma, \delta, \varrho, \nu, \xi, \chi, \varepsilon, \zeta, \eta) :$$

$$c_{\tilde{p}} = g^\alpha h^\beta \ \wedge \ c_{\tilde{q}} = g^\gamma h^\delta \ \wedge \ c_p = g^\varrho h^\nu \ \wedge \ c_q = g^\xi h^\chi \ \wedge \tag{27}$$

$$c_p/(c_{\tilde{p}}^2 g) = h^\varepsilon \ \wedge \ c_q/(c_{\tilde{q}}^2 g) = h^\zeta \ \wedge \ c_n = c_p^\xi h^\eta \ \wedge \tag{28}$$

$$\alpha \in \texttt{primes}(t) \ \wedge \ \gamma \in \texttt{primes}(t) \wedge \tag{29}$$

$$\varrho \in \texttt{primes}(t) \ \wedge \ \xi \in \texttt{primes}(t)\} \ , \tag{30}$$

where t denotes the number of bases used in Lehmann's primality tests. (The length conditions on α, γ, ϱ, and ξ are shown in the $\texttt{primes}(t)$-parts of the protocol.)

Theorem 5. *Assume computing discrete logarithms in G is infeasible. Then, the above protocol is a statistical zero-knowledge argument that the integer committed to by c_n is the product of product of two integers p and q and p, q, $(p-1)/2$ and $(q-1)/2$ are primes. The soundness error probability is at most $2^{-k} + 2^{-t}$.*

Proof. The proof is straight forward from the Theorems 1, 2, and 4. □

The computational and communication costs of this protocol are reigned by the primality-protocols and thus about four times as high as for a single primality-protocol (cf. Subsection 4.3).

5.2 A Protocol for a Publicly Known RSA Modulus

In cases the number n is publicly known and fulfills some side-conditions (see below), much less rounds of the Lehmann test will be sufficient if the prover and the verifier first run the protocol due to Gennaro et al. [22] (which includes the protocols proposed by Peralta & van de Graaf [37] and by Boyar et al. [2]). This protocol is a statistical zero-knowledge proof system that there exist two integers $a, b \geq 1$ such that n consists of two primes $p = 2\tilde{p}^a + 1$ and $q = 2\tilde{q}^b + 1$ with $p, q, \tilde{p}, \tilde{q} \not\equiv 1 \pmod 8$, $p \not\equiv q \pmod 8$, $\tilde{p} \not\equiv \tilde{q} \pmod 8$ and $\tilde{p}, \tilde{q}$ are primes. Given the fact that $(p-1)/2$ is a prime power, and assuming that it is not prime, the probability that it passes a single round of Lehmann's primality test for any $a > 1$ is at most $\tilde{p}^{1-a} \leq \sqrt{2/(p-1)}$ (for q the corresponding statement hold). Hence, if p and q are sufficiently large, a single round of Lehmann's primality test on $(p-1)/2$ and $(q-1)/2$ will be sufficient to prove their primality with overwhelming probability. Thus, the resulting protocol to prove that n is the product of two safe primes is as follows.

1. First the prover computes $c_p := g^p h^{r_p}$, $c_{\tilde{p}} := g^{(p-1)/2} h^{r_{\tilde{p}}}$, $c_q := g^q h^{r_b}$, and $c_{\tilde{q}} := g^{(q-1)/2} h^{r_{\tilde{p}}}$ with $r_p, r_{\tilde{p}}, r_q, r_{\tilde{q}} \in_R \mathbb{Z}_Q$ and sends these commitments together with n to the verifier.
2. The prover and the verifier carry out the protocol by Gennaro et al. [22]
3. and then k times the protocol denoted by

$$S_{52} := PK\{(\alpha, \beta, \gamma, \delta, \varrho, \epsilon, \xi, \chi, \varepsilon, \zeta, \eta) :$$
$$c_{\tilde{p}} = g^\alpha h^\beta \ \wedge \ c_{\tilde{q}} = g^\gamma h^\delta \ \wedge \ c_p = g^\varrho h^\epsilon \ \wedge \ c_q = g^\xi h^\chi \ \wedge \tag{31}$$
$$c_p/(c_{\tilde{p}}^2 g) = h^\varepsilon \ \wedge \ c_q/(c_{\tilde{q}}^2 g) = h^\zeta \ \wedge \ g^n = c_p^\xi h^\eta \ \wedge \tag{32}$$
$$\alpha \in \texttt{primes}(1) \ \wedge \ \gamma \in \texttt{primes}(1)\} \ , \tag{33}$$

where the length conditions on α and γ are hidden within in the sub-protocols $\texttt{primes}(1)$.

Theorem 6. *Let n be the product of two primes p and q such that $p = 2\tilde{p}^a + 1$ and $q = 2\tilde{q}^b + 1$ with $p, q, \tilde{p}, \tilde{q} \not\equiv 1 \pmod 8$, $p \not\equiv q \pmod 8$, $\tilde{p} \not\equiv \tilde{q} \pmod 8$, $a, b \geq 1$ and $\tilde{p}, \tilde{q}$ are primes. Assume computing discrete logarithms in G is infeasible. Then, the protocol S_{52} is a statistical zero-knowledge argument that n is the product of two integers p and q and that p, q, $(p-1)/2$, and $(q-1)/2$ are primes. Assume $p > q$. Then the soundness error probability is at most $2^{-k} + \sqrt{2/(q-1)}$.*

The computational and communication costs of this protocol is dominated by the costs of a single round (i.e., $t = 1$) of the primality protocol described in the previous section and the costs of protocol of Gennaro et al. [22].

It is obvious how to apply our techniques to get a protocol for proving that n is the product of two *strong* primes [24] (i.e., $(p-1)/2$, $(q-1)/2$, $(p+1)/2$ and $(q+1)/2$ are primes or have a large prime factor) or, more general, two primes p and q such that $\Phi_k(p)$ and $\Phi_k(q)$ are not smooth, where Φ_k is the k-th cyclotomic polynomial. (Recall that smoothness of $\Phi_k(p)$ or $\Phi_k(q)$ for any integer $k > 0$, $k = O(\log n)$ allows to factor n efficiently [1]). Lower bounds on p, q, and on n might also be shown. Also, factors r other than 2 in $(p-1)/r$ could easily be incorporated.

6 Acknowledgments

The authors are grateful to Christian Cachin, Ivan Damgård, Markus Stadler, and Robert Zuccherato for valuable discussions and helpful comments. While the second author was with Ubilab, UBS, he was supported by the Swiss National Foundation (SNF/SPP project no. 5003-045293).

References

1. E. Bach and J. Shallit. Factoring with cyclotomic polynomials. In *26th FOCS*, IEEE, pp. 443–450, 1985.

2. J. Boyar, K. Friedl, and C. Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. *Journal of Cryptology*, 4(3):185–206, 1991.
3. S. Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology — CRYPTO '93*, volume 773 of *LNCS*, pp. 302–318, 1993.
4. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pp. 318–333. Springer Verlag, 1997.
5. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, Oct. 1988.
6. J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. Technical Report RS-98-29, BRICS, Departement of Computer Science, University of Aarhus, Nov. 1998.
7. J. Camenisch and M. Michels. A group signature scheme based on an RSA-variant. Tech. Rep. RS-98-27, BRICS, Departement of Computer Science, University of Aarhus, Nov. 1998. Preliminary version appeared in *Advances in Cryptology — ASIACRYPT '98*, volume 1514 of *LNCS*, pages 160–174. Springer Verlag, 1998.
8. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97*, volume 1296 of *LNCS*, pp. 410–424. Springer Verlag, 1997.
9. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich, Mar. 1997.
10. J. L. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998. Diss. ETH No. 12520.
11. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pp. 561–575. Springer Verlag, 1998. Revised version available as GTE Technical Report.
12. D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology — EUROCRYPT '87*, volume 304 of *LNCS*, pp. 127–141. Springer-Verlag, 1988.
13. D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In *Advances in Cryptology — CRYPTO '86*, volume 263 of *LNCS*, pp. 200–212. Springer-Verlag, 1987.
14. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology — CRYPTO '92*, volume 740 of *LNCS*, pp. 89–105. Springer-Verlag, 1993.
15. H. Cohen. *A Course in Computational Algebraic Number Theory*. Number 138 in Graduate Texts in Mathematics. Springer-Verlag, Berlin, 1993.
16. R. Cramer and I. Damgård. Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? In *Advances in Cryptology — CRYPTO '98*, volume 1642 of *LNCS*, pp. 424–441, Berlin, 1998. Springer Verlag.
17. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94*, volume 839 of *LNCS*, pp. 174–187. Springer Verlag, 1994.
18. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — CRYPTO '97*, volume 1294 of *LNCS*, pp. 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 *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pp. 32–46. Springer Verlag, 1998.

20. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *Advances in Cryptology — CRYPT0 '96*, volume 1109 of *LNCS*, pp. 157–172, Berlin, 1996. IACR, Springer Verlag.
21. R. Gennaro, H. Krawczyk, and T. Rabin. RSA-based undeniable signatures. In *Advances in Cryptology — CRYPTO '97*, volume 1296 of *LNCS*, pp. 132–149. Springer Verlag, 1997.
22. R. Gennaro, D. Micciancio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *5rd ACM Conference on Computer and Communicatons Security*, 1998.
23. O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design. In *Advances in Cryptology — CRYPTO '86*, volume 263 of *LNCS*, pp. 171–185. Springer-Verlag, 1987.
24. J. Gordon. Strong RSA keys. *Electronics Letters*, 20(12):514–516, 1984.
25. K. Koyama, U. Maurer, T. Okamoto, and S. Vanstone. New public-key schemes based on elliptic curves over the ring Z_n. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *LNCS*, pp. 252–266. Springer-Verlag, 1992.
26. E. Kranakis. *Primality and Cryptography.* Wiley-Teubner Series in Computer Science, 1986.
27. D. J. Lehmann. On primality tests. *SIAM Journal of Computing*, 11(2):374–375, May 1982.
28. M. Liskov and B. Silverman. A Statisical limited-knowledge proof for secure RSA keys. manuscript, (1998).
29. W. Mao. Verifable Partial Sharing of Integer Factors. to appear in Proc. *SAC*'98, 1998.
30. G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.
31. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *LNCS*, pp. 129–140. Springer Verlag, 1992.
32. J. M. Pollard. Theorems on factorization and primality testing. *Proc. Cambridge Philosophical Society*, 76:521–528, 1974.
33. M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.
34. A. de Santis, L. di Crescenzo, G. Persiano, M. Yung. On Monotone Formula Closure of SZK. *35th FOCS*, IEEE, pp. 454–465, 1994.
35. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
36. R. Solovay and V. Strassen. A fast monte-carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, Mar. 1977.
37. J. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In *Advances in Cryptology — CRYPTO '87*, volume 293 of *LNCS*, pp. 128–134. Springer-Verlag, 1988.
38. H. C. Williams. A $p + 1$ method of factoring. *Mathematics of Computation*, 39(159):225–234, 1982.
39. X9.31 - 1998 Digital Signatures using reversible public key cryptography for the financial services industry (rDSA). *American National Standard, Working Draft*, 59 pages, 1998.