Basic Research in Computer Science

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

Jan Camenisch
Markus Michels

BRICS Report Series **RS-98-29**

ISSN 0909-0878 **November 1998**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/98/29/`

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

Jan Camenisch

BRICS*
Department of Computer Science
University of Aarhus
DK – 8000 Århus C, Denmark
camenisch@daimi.au.dk

Markus Michels†

Entrust Technologies Europe
r3 security engineering ag
Glatt Tower
CH – 8301 Glattzentrum, Switzerland
Markus.Michels@entrust.com

November, 1998

Abstract

This paper presents the first efficient statistical zero-knowledge protocols to prove statements such as:

- A committed number is a pseudo-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 primes as well.
- A given value is of large order modulo a composite number that consists of two safe prime factors.

So far, no methods other than inefficient circuit-based proofs are known for proving such properties. Proving the second property is for instance necessary in many recent cryptographic schemes that rely on both the hardness of computing discrete logarithms and of difficulty computing roots modulo a composite.

The main building blocks of our protocols are statistical zero-knowledge proofs that are of independent interest. Mainly, we show how to prove the correct computation of a modular addition, a modular multiplication, or a modular exponentiation, where all values including the modulus are committed but *not* publicly known. Apart from the validity of the computation, no other information about the modulus (e.g., a generator which order equals the modulus) or any other operand is given. Our technique can be generalized to prove in zero-knowledge that any multivariate polynomial equation modulo a certain modulus is satisfied, where only commitments to the variables of the polynomial and a commitment to the modulus must be known. This improves previous results, where the modulus is publicly known.

We show how a prover can use these building blocks to convince a verifier that a committed number is prime. This finally leads to efficient protocols for

*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.

proving that a committed (or revealed) number is the product of two safe primes. As a consequence, it can be shown that a given value is of large order modulo a given number that is a product of two safe primes.

Keywords. RSA-based protocols, zero-knowledge proofs of knowledge, primality tests.

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 recent cryptographic schemes (e.g., [7, 18, 19]) whose security is based on the infeasibility of computing discrete logarithms and of computing roots in groups of unknown order. In such scheme there typically is a designated entity which knows the group's order and hence can compute roots. Although the other entities must not learn the group's order, they still want to be assured that the order is not smooth, since that would allow the designated entity to compute discrete logarithms. One example of such a group are subgroups of $\mathbb{Z}_n^*$. In this case, it suffices that the designated entity proves that n is the product of two safe primes, i.e., primes p and q such that $(p-1)/2$ and $(q-1)/2$ are primes as well [19]. An other example of such a group are elliptic curves over $\mathbb{Z}_n$. There, n must be the product of two primes p and q such that $(p+1)/2$ and $(q+1)/2$ are also primes [23]. 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[1]. Previously, the only way known for proving such properties was applying inefficient general zero-knowledge proof techniques (e.g., [21, 6, 14]).

Our main results are as follows: First, we provide an efficient protocol to prove 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, to any multivariate polynomial. Previous protocols allow only to prove algebraic relations modulo a *publicly known* integer [5, 8, 16, 14] were known. Our schemes work also for the class of commitments described in [14] (that includes discrete-logarithm-based and RSA-based commitment schemes). Second, we present an efficient zero-knowledge proof for pseudo-primality of a committed number and, as a consequence, a zero-knowledge proof 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 modulus n is publicly known, however, more efficient protocols can be obtained by combining our techniques with known results described in the next paragraph.

Based on the these proofs it is simple to show that a given element $a \in \mathbb{Z}_n^*$ has a large order modulo a given $n = pq$ when $(p-1)/2$ and $(q-1)/2$ are primes. First the prover shows that n is indeed of this form. Then the verifier checks whether $a^2 \not\equiv 1$

[1] It should be mentioned, however, that it is unnecessary to add this requirement into the RSA key generation explicitly. 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 to guarantee that the Pollard-Rho and Williams p+1 factoring methods [28, 33] do not work. On the other hand, a proof that an arbitrarily generated RSA modulus is not weak without revealing the prime factors seems to be hard to obtain, as an infinite number of conditions have to be checked (e.g., see [1]).

$(\mathrm{mod}\ n)$ and $\gcd(a^2-1,n)=1$ holds. From this it follows that a can only be of order $(p-1)(q-1)/4$ or $(p-1)(q-1)/2$.

Let us finally summarize related results on proving properties of composite numbers. Van de Graaf and Peralta [32] provide an efficient proof that a given modulus n is of the form $n=p^rq^s$, 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. [3] allows to prove that a given n is square-free, i.e., there is no prime p with $p|n$ such that $p^2|n$. Hence, if for a given n both properties can be shown, it follows that n is of form $n=pq$, where p and q are primes and $p\equiv q\equiv 3 \pmod 4$. This result was recently strengthened by Gennaro et al. [20] 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 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. Let us further mention the work of Boneh and Franklin [2], who provide a proof that a distributively generated number n indeed consists of two primes (without further showing that these primes are of special form). It should be noted that all these solutions assume that n is publicly known.

2 Tools

2.1 Commitment Schemes

Our schemes build use commitment schemes that allow to algebraic prove properties of the committed value. There are two kinds of commitment scheme. 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 zero-knowledge arguments (proofs of knowledge) or be zero-knowledge proof systems.

Cramer and Damgård [14] describe a class of commitment schemes allowing to prove algebraic properties of the committed value. These include RSA-based and discrete-logarithm-based schemes for both kinds of commitment scheme. An example of a computationally binding and unconditionally hiding scheme based on the discrete logarithm problem is the one to Pedersen [27]. Given are a group G of prime order Q and two random generators g and h such that $\log_g h$ is unknown and computing discrete logarithms is infeasible. A value $a\in\mathbb{Z}_Q$ is committed to as $c_a := g^a h^r$, where r is randomly chosen from $\mathbb{Z}_Q$. For easier description, we will use this commitment scheme for our protocols and hence they will be statistical zero-knowledge proofs of knowledge. However, the protocol can easily be adapted to work for all the commitment scheme exposed in [14].

2.2 Various Proof-Protocols Found in Literature

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, i.e., discrete logarithms are allowed to be negative.

We shortly review various systems for proving knowledge of and about discrete logarithms found in literature.

Proving the knowledge of a discrete logarithm x of a group element y to a basis g [11, 30]. 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, iff $g^s y^c = t$ holds. This protocol is an *honest-verifier zero-knowledge proof of knowledge* for $k = \Theta(\text{poly}(\log Q))$ and a *zero-knowledge proof of knowledge* for $k = \mathcal{O}(\log\log(Q))$ 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 [7], we denote this protocol by $PK\{(\alpha) : y = g^\alpha\}$, where *PK* stands for "proof of knowledge".

Proving the knowledge of a representation of the element y to the bases $g_1, \ldots, g_l$ [4, 10], 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 to multiple bases. The prover chooses random $r_1, \ldots, r_l \in_R \mathbb{Z}_Q$, computes $t := \prod_{i=1}^{l} g_i^{r_i}$, 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 $s_i := r_i - cx_i \pmod{Q}$ for $i = 1, \ldots, l$ and sends all s_i's to the verifier. The verifier accepts, iff $t = y^c \prod_{i=1}^{l} g_i^{s_i}$ holds. This protocol is denoted $PK\{(\alpha_1, \ldots, \alpha_l) : y = \prod_{i=1}^{l} g_i^{\alpha_i}\}$.

Proving the equality of the discrete logarithms of the elements y_1 and y_2 to the bases g and h, respectively [12]. 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, iff $g^s y_1^c = t_1$ 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) [15]. W.l.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 $s_1 := r_1 - c_1 x \pmod{Q}$ and sends s_1, s_2, c_1, and c_2 to the verifier. The verifier accepts, iff $c_1 \oplus c_2 = c$ and $t_i = g^{s_i} y_i^{c_i}$ holds for $i \in \{1,2\}$. This protocol is denoted $PK\{(\alpha, \beta) : y_1 = g^\alpha \vee y_2 = g^\beta\}$. In their paper [15], Cramer et al. generalize this approach to an efficient system for proving arbitrary monotone statements built with $\wedge$'s and $\vee$'s.

Proving that a discrete logarithm lies in a given range. The last building block for our protocols are statistical zero-knowledge proofs that the discrete logarithm x

of y to the base g satisfies $2^{\ell_1} - 2^{\ell_2} < x < 2^{\ell_1} + 2^{\ell_2}$ for given parameter ℓ_1 and ℓ_2. The parameter 2^{ℓ_1} acts as an offset and can also chosen to be zero. In principle, such a proof can be given by committing to every bit of x and proving that the committed values are indeed 0's or 1's and that they are the binary representation of x. Fortunately, there is a much more efficient way to achieve this as is shown in [9, 16]. The price one has to pay is that, first, the protocol is only statistical zero-knowledge and, second, it can only be shown that $2^{\ell_1} - 2^{\epsilon\ell_2+2} < x < 2^{\ell_1} + 2^{\epsilon\ell_2+2}$, where $\epsilon > 1$ is a security parameter, although x must lie in the intervals $2^{\ell_1} - 2^{\ell_2} < x < 2^{\ell_1} + 2^{\ell_2}$ for the prover being able to successfully carry out the proof. Finally, if the group's order is known, only binary challenges are possible. Since the protocol is not so well known, we describe it in full detail in Appendix A. The protocol is denoted 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).

It should be mentioned, however, that 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] then larger challenges can be chosen [16, 17]. Although we describe our protocols for the setting where the group's order is known to the prover, all protocols can easily be adapted to the setting where the prover does not know the group's order using the techniques from [16, 17].

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 above proofs. 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 [15]. In the following we will use of such compositions without having explained the technical details for composition for which we refer to [5, 8, 15].

3 Secret Computations with a Secret Modulus

In this section we assume that a prover has committed to some integers a, b, d, and n. We will provide an efficient protocol for proving that $a^b \equiv d \pmod{n}$ holds for the committed integers without revealing any further information to the verifier (i.e., the proof is zero-knowledge). However, before we can do so, we need protocols to prove that a committed integer is the addition or the multiplication of two committed secret integers modulo a committed secret modulus 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 parameters (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

[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 $z \equiv u^e \pmod{n}$.

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 $\mathbb{Z}_Q$.

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 running the protocol denoted[3]:

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

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

$$\begin{aligned} S_* := PK\big\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\eta,\vartheta,\varkappa,\lambda,\mu,\xi,\rho,\sigma) : \\ c_a = g^\alpha h^\beta \ \wedge\ -2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}} \ \wedge\ c_b = g^\gamma h^\delta \ \wedge\ -2^{\ddot{\ell}} < \gamma < 2^{\ddot{\ell}} \ \wedge \\ c_d = g^\varepsilon h^\zeta \ \wedge\ -2^{\ddot{\ell}} < \varepsilon < 2^{\ddot{\ell}} \ \wedge\ c_n = g^\eta h^\vartheta \ \wedge\ -2^{\ddot{\ell}} < \eta < 2^{\ddot{\ell}} \ \wedge \\ c_d = c_b^\alpha c_n^\rho h^\sigma \ \wedge\ -2^{\ddot{\ell}} < \rho < 2^{\ddot{\ell}}\big\} \end{aligned}$$

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. Then the protocol S_+ is a statistical zero-knowledge proof that $a + b \equiv d \pmod{n}$ holds. Furthermore, the protocol S_* is a statistical zero-knowledge proof that $ab \equiv d \pmod{n}$ holds.*

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

Let us argue why the modular relations 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.

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

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 that the discrete log of h to the base g 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 hence

$$\hat{d} \equiv \hat{a} + \hat{b} \pmod{\hat{n}}$$

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 $c_d = c_b^{\hat{a}} c_n^{\hat{v}} h^{r_6}$ and thus $g^{\hat{d}}h^{\hat{r}_3} = g^{\hat{a}\hat{b}+\hat{v}\hat{n}}h^{\hat{a}\hat{r}_2+\hat{v}\hat{r}_4+\hat{r}_*}$ holds. Again, provided that the discrete logarithm of h to the base g is not known, we have

$$\hat{d} \equiv \hat{a}\hat{b} + \hat{v}\hat{n} \pmod{Q}.$$

As before, because of $2^{2\ddot{\ell}+1} < Q$ and the constraints on 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 given in 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 an integer $\tilde{a}$ that equals a^b (in $\mathbb{Z}$) and proving this fact, would required that G has order about $2^{b\ell}$ and thus such a proof would become rather inefficient.

Below we expose a more efficient protocol for proving this which is obtained by constructing $a^b \pmod{n}$ step by step according to the square & multiply algorithm (cf. Appendix B for easy reference). (In practice a more enhanced exponentiation algorithm might be used (see, e.g., [13]), but one should keep in mind that it must not leak additional information about the exponent.) In the following, we assume that an upper-bound $\ell_b \leq \ell$ on the length of b is publicly known.

1. Apart from committing to a, $b = \sum_{i=0}^{\ell_b-1} b_i 2^i$, d, and n the prover must also commit to all 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 $c_{\nu_i} := g^{(a^{2^i} \pmod{n})}h^{\hat{r}_i}$, $(i = 1, \ldots, \ell_b - 1)$, 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 $\bar{r}_i \in_R \mathbb{Z}_Q$.

2. To prove that $a^b \equiv d \pmod{n}$ holds, the prover sends all her commitments to the verifier and then they carry out the protocol

$$
\begin{aligned}
S_{exp} := PK\Big\{ & (\alpha, \beta, \xi, \chi, \gamma, \delta, \varepsilon, \zeta, \eta, (\lambda_i, \mu_i, \nu_i, \xi_i, \sigma_i, \tau_i, \vartheta_i, \varphi_i, \psi_i)_{i=1}^{\ell_b - 1}, (\varkappa_i, \rho_i)_{i=1}^{\ell_b - 2},) : \\
& c_a = g^{\alpha} h^{\beta} \ \wedge -2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}} \ \wedge && (1) \\
& c_d = g^{\gamma} h^{\delta} \ \wedge -2^{\ddot{\ell}} < \gamma < 2^{\ddot{\ell}} \ \wedge && (2) \\
& c_n = g^{\varepsilon} h^{\zeta} \ \wedge -2^{\ddot{\ell}} < \varepsilon < 2^{\ddot{\ell}} \ \wedge && (3) \\
& \Big(\prod_{i=0}^{\ell_b - 1} c_{b_i}^{2^i}\Big) / c_b = h^{\eta} \ \wedge && (4) \\
& c_{\nu_1} = g^{\lambda_1} h^{\mu_1} \ \wedge \ \ldots \ \wedge \ c_{\nu_{\ell_b - 1}} = g^{\lambda_{\ell_b - 1}} h^{\mu_{\ell_b - 1}} \ \wedge && (5) \\
& c_{\nu_1} = c_a^{\alpha} c_n^{\nu_1} h^{\xi_1} \ \wedge \ c_{\nu_2} = c_{\nu_1}^{\lambda_1} c_n^{\nu_2} h^{\xi_2} \ \wedge \ \ldots \wedge \ c_{\nu_{\ell_b - 1}} = c_{\nu_{\ell_b - 2}}^{\lambda_{\ell_b - 2}} c_n^{\nu_{\ell_b - 1}} h^{\xi_{\ell_b - 1}} \ \wedge && (6) \\
& -2^{\ddot{\ell}} < \lambda_1 < 2^{\ddot{\ell}} \ \wedge \ldots \wedge -2^{\ddot{\ell}} < \lambda_{\ell_b - 1} < 2^{\ddot{\ell}} \ \wedge && (7) \\
& -2^{\ddot{\ell}} < \nu_1 < 2^{\ddot{\ell}} \ \wedge \ldots \wedge -2^{\ddot{\ell}} < \nu_{\ell_b - 1} < 2^{\ddot{\ell}} \ \wedge && (8) \\
& c_{u_1} = g^{\varkappa_1} h^{\rho_1} \ \wedge \ \ldots \ \wedge \ c_{u_{\ell_b - 2}} = g^{\varkappa_{\ell_b - 2}} h^{\rho_{\ell_b - 2}} \ \wedge && (9) \\
& -2^{\ddot{\ell}} < \varkappa_1 < 2^{\ddot{\ell}} \ \wedge \ldots \wedge -2^{\ddot{\ell}} < \varkappa_{\ell_b - 2} < 2^{\ddot{\ell}} \ \wedge && (10) \\
& \Big(\big(c_{b_0} = h^{\sigma_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 && (11) \\
& \Big(\big(c_{b_1} = h^{\sigma_1} \ \wedge \ c_{u_1} / c_{u_0} = h^{\tau_1}\big) \vee && (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^{\sigma_{\ell_b - 2}} \ \wedge \ c_{u_{\ell_b - 2}} / c_{u_{\ell_b - 3}} = h^{\tau_i}\big) \vee && (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^{\sigma_{\ell_b - 1}} \ \wedge \ c_d / c_{u_{\ell_b - 2}} = h^{\tau_i}\big) \vee && (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\} .
\end{aligned}
$$

Let us now explain why this protocol proves that $a^b \equiv d \pmod{n}$ holds and consider the clauses of sub-protocol S_{exp}. 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 that 2^{ℓ_b}. The Clauses 5–8 prove that the c_{ν_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 $i = 1$ to $\ell_b - 2$ 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 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* a, b, d, *and* n *be integers that are committed in* c_a, c_b, c_d, *and* c_n *by the prover 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 her auxiliary commitments. Then the protocol* S_{exp} *is a statistical zero-knowledge proof that the equation* $a^b \equiv d \pmod{n}$ *holds.*

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 will abbreviate the protocol S_{exp} by a clause like $\left(\alpha^\beta \equiv \gamma \pmod{\delta}\right)$ to the statement that is proven and assume that the prover send 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

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 the size of 10 group elements and $5\epsilon\ell$ bits in case of S_+ and about the size of 11 group elements and $5\epsilon\ell$ bits in case of S_*.

In case of the exponentiation proof the verifier and the prover need to compute about $6\ell_b$ multi-exponentiations per round, while the prover needs to compute about $3\ell_b$ multi-exponentiations for the commitments to the intermediary results of the square & multiply algorithm. The communication cost per round is about the size of $12\ell_b$ group elements and $4\ell_b\epsilon\ell$ 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

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

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 can be shown: 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 modular exponentiation proof described above and that for all i the same x_j is in $p_{i,j}$. Clearly, several such polynomials can be combined as well.

4 A Proof That a Secret Number is a Pseudo-Prime

In this section we describe how the prover and the verifier can carry out a primality test for an integer that is only given by a commitment. Some primality tests reveal information about the structure of the prime and are hence not suited unless one is willing to give away this information. Examples of such tests are the Miller-Rabin test [26, 29] or the one based on Pocklington's theorem. A test that does not reveal such information is the one due to Lehmann [25] which we describe in the next subsection.

4.1 Lehmann's Primality Test

Lehmann's primality test is variation of the Solovay-Strassen [31] primality test and based on the following theorem [24]:

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

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

This theorem suggest the following probabilistic primality test:

- 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 whether $a_i^{(n-1)/2} \equiv -1 \pmod{n}$ holds for at least one i.

The probability that a non-prime n passes this test is at most 2^{-k}. 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 the Lehmann-test is equivalent to the Miller-Rabin test and the failure probability is at most 4^{-k} [29].

4.2 Proving the Pseudo-Primality of a Committed Number

We now show how the prover and the verifier can do Lehmann's primality test for a number committed by 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 prove that for these bases $a_i^{(n-1)/2} \equiv \pm 1 \pmod{n}$ holds. Furthermore, the prover must commit to a base, say $\tilde{a}$, such that $\tilde{a}^{(n-1)/2} \equiv -1 \pmod{n}$ holds to satisfy the second condition in Theorem 3.

Let ℓ be an integer such that $n < 2^\ell$ holds and let $\epsilon > 1$ be 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

$$
\begin{aligned}
S_p := PK\Big\{ & (\alpha, \beta, \gamma, \nu, \xi, \rho, \kappa, (\delta_i, \varepsilon_i, \zeta_i, \eta_i, \vartheta_i, \varkappa_i, \rho_i, \kappa_i, \mu_i, \psi_i)_{i=1}^{t} : \\
& c_b = g^{\alpha} h^{\beta} \ \wedge \ -2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}} \ \wedge && (15) \\
& c_n = g^{\nu} h^{\xi} \ \wedge \ -2^{\ddot{\ell}} < \nu < 2^{\ddot{\ell}} \ \wedge && (16) \\
& c_b^2 g / c_n = h^{\gamma} \ \wedge && (17) \\
& c_{\tilde{a}} = g^{\rho} h^{\kappa} \ \wedge \ (\rho^{\alpha} \equiv -1 \pmod{\nu}) \ \wedge && (18) \\
& c_{\hat{a}_1} = g^{\delta_1} h^{\varepsilon_1} \ \wedge \ \ldots \wedge \ c_{\hat{a}_t} = g^{\delta_t} h^{\varepsilon_t} \ \wedge && (19) \\
& c_{a_1} / g^{\check{a}_1} = g^{\delta_1} c_n^{\zeta_1} h^{\eta_1} \ \wedge \ \ldots \wedge \ c_{a_t} / g^{\check{a}_t} = g^{\delta_t} c_n^{\zeta_t} h^{\eta_t} \ \wedge && (20) \\
& -2^{\ddot{\ell}} < \delta_1 < 2^{\ddot{\ell}} \ \wedge \ \ldots \wedge \ -2^{\ddot{\ell}} < \delta_t < 2^{\ddot{\ell}} \ \wedge && (21) \\
& -2^{\ddot{\ell}} < \zeta_1 < 2^{\ddot{\ell}} \ \wedge \ \ldots \wedge \ -2^{\ddot{\ell}} < \zeta_t < 2^{\ddot{\ell}} \ \wedge && (22) \\
& c_{a_1} = g^{\rho_1} h^{\kappa_1} \ \wedge \ \ldots \wedge \ c_{a_t} = g^{\rho_t} h^{\kappa_t} \ \wedge && (23) \\
& \left(c_{d_1} / g = h^{\vartheta_1} \vee c_{d_1} g = h^{\vartheta_1}\right) \ \wedge \ \ldots \wedge \ \left(c_{d_t} / g = h^{\vartheta_t} \vee c_{d_t} g = h^{\vartheta_t}\right) \ \wedge && (24) \\
& c_{d_1} = g^{\mu_1} h^{\psi_1} \ \wedge \ \ldots \wedge \ c_{d_t} = g^{\mu_t} h^{\psi_t} \ \wedge && (25) \\
& (\rho_1^{\alpha} \equiv \mu_1 \pmod{\nu}) \ \wedge \ \ldots \wedge \ (\rho_t^{\alpha} \equiv \mu_t \pmod{\nu}) \Big\} . && (26)
\end{aligned}
$$

This concludes 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, 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_{exp}.

Theorem 4. *Given a commitment c_n to an integer, the above protocol is a statistical zero-knowledge proof that the committed integer is a prime with error-probability at most 2^{-t} for the primality-test.*

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

Similar as for modular exponentiation we will abbreviate the above protocol by adding a clause such as $\alpha \in \texttt{pseudoprimes}(t)$ 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 that, she can additionally prove that she knows χ and ψ such that $c_b/g = (g^2)^\chi h^\psi$ and $-2^{\ell'} < \chi < 2^{\ell'}$ 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 $t + 1$ proofs that a modular exponentiation holds are needed where the exponents are about $\log n$ bits. Thus, the verifier needs to compute about $6t \log n$ multi-exponentiations per round and the prover needs to compute about $2t \log n$ multi-exponentiations for the commitments to the intermediary results of the square & multiply algorithm. The communication cost per round is about the size of $12t \log n$ group elements and $4t \log n \epsilon \ell$ bits and an initial $2t \log n$ group element which are the commitments to the intermediary results of the square & multiply algorithm and the commitments to the bases for 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 the modulus is not known to the verifier, i.e., only a commitment of the modulus is given. 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 an integer n, where she choose $r_n \in_R \mathbb{Z}_Q$ 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 (pseudo-)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 carry out the following protocol

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

$$c_{\tilde{p}} = g^{\alpha} h^{\beta} \ \wedge \ (-2^{\ddot{\ell}} < \alpha < 2^{\ddot{\ell}}) \ \wedge \tag{27}$$

$$c_{\tilde{q}} = g^{\gamma} h^{\delta} \ \wedge \ (-2^{\ddot{\ell}} < \delta < 2^{\ddot{\ell}}) \ \wedge \tag{28}$$

$$c_p = g^{\rho} h^{\nu} \ \wedge \ c_q = g^{\xi} h^{\chi} \ \wedge \tag{29}$$

$$c_p / (c_{\tilde{p}}^2 g) = h^{\varepsilon} \ \wedge \ c_q / (c_{\tilde{q}}^2 g) = h^{\zeta} \ \wedge \ c_n / (c_p c_q) = h^{\eta} \ \wedge \tag{30}$$

$$\alpha \in \texttt{pseudoprimes}(t) \ \wedge \ \gamma \in \texttt{pseudoprimes}(t) \wedge \tag{31}$$

$$\rho \in \texttt{pseudoprimes}(t) \ \wedge \ \xi \in \texttt{pseudoprimes}(t)\}\,, \tag{32}$$

where t denotes the number of bases used in the Lehmann-primality tests.

Theorem 5. *Let n be an integer that is committed by c_n. Then the above protocol is a statistical zero-knowledge proof that n is an RSA modulus of form $n = pq$ where $p, q, (p-1)/2$ and $(q-1)/2$ are primes with error-probability at most 2^{-t} each of for the primality tests.*

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

The efficiency is reigned by the (pseudo-)primality-proofs and thus about four times as high as for a single (pseudo-)primality-proof (cf. Subsection 4.3).

5.2 A Protocol For a Publicly Known RSA Modulus

We now consider the case where the modulus n is publicly known. In case n fulfils certain side-conditions (see below), it is more efficient to first run the protocol due to Gennaro et al. [20] (which includes the proofs proposed by Peralta & van de Graaf [32] and by Boyar et al. [3]). 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 $p, q, \tilde{p}, \tilde{q} \not\equiv 1 \pmod 8$, $p \not\equiv q \pmod 8$, and $\tilde{p} \not\equiv \tilde{q} \pmod 8$. Given the fact that $(p-1)/2$ and $(p-1)/2$ are prime powers, the probability that they pass a single round of the Lehmann's primality test for any $a > 1$ and $b > 1$, is at most $\tilde{p}^{1-a} \leq \sqrt{2/(p-1)}$ and $\tilde{q}^{1-a} \leq \sqrt{2/(q-1)}$, respectively, if they are not prime. Hence, if p and q are sufficiently large, a single round of the Lehmann-primality test on $(p-1)/2$ and $(q-1)/2$ will be sufficient to prove their (pseudo-)primality.

We now describe the protocol that allows the prover to prove the verifier that a given integer n is the product of two safe (pseudo-)primes.

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. [20]

3. and then the protocol denoted

$$S_{52} := PK\{(\alpha, \beta, \gamma, \delta, \rho, \epsilon, \xi, \chi, \varepsilon, \zeta, \eta) :$$
$$c_{\tilde{p}} = g^\alpha h^\beta \ \wedge \ (-2^{\ddot{\ell}/2} < \alpha < 2^{\ddot{\ell}/2}) \ \wedge \ c_{\tilde{q}} = g^\gamma h^\delta \ \wedge \ (-2^{\ddot{\ell}/2} < \delta < 2^{\ddot{\ell}/2}) \ \wedge \tag{33}$$
$$c_p = g^\rho h^\epsilon \ \wedge \ c_q = g^\xi h^\chi \ \wedge \ c_p/(c_{\tilde{p}}^2 g) = h^\varepsilon \ \wedge \ c_q/(c_{\tilde{q}}^2 g) = h^\zeta \ \wedge \tag{34}$$
$$g^n/(c_p c_q) = h^\eta \ \wedge \ \gamma \in \texttt{pseudoprimes}(1) \ \wedge \ \alpha \in \texttt{pseudoprimes}(1)\} . \tag{35}$$

Theorem 6. *Let $n = pq$ be a given integer that passes the test given in [20] with error probability at most 2^{-z} for an integer $z \geq 1$. Then the above protocol is a statistical zero-knowledge proof that n is an RSA modulus of form $n = pq$ where $p, q, (p-1)/2$ and $(q-1)/2$ are primes with error probability at most $1 - (1 - 2^{-z})(1 - \sqrt{2/(p-1)})(1 - \sqrt{2/(q-1)}) < 2^{-z} + \sqrt{2/(p-1)} + \sqrt{2/(q-1)} + 2^{-z}\sqrt{2/(p-1)}\sqrt{2/(q-1)}$.*

The efficiency for this protocol is dominated by the efficiency of a single round (i.e., $t = 1$) of the (pseudo-)primality proof described in the previous section and the efficiency of protocol of Gennaro et al. [20].

6 Conclusion

We have presented efficient protocols for proving that modular relations among secret values (including the modulus!) hold and for proving that an given (or only committed-to) number is the product of two safe primes.

We note that it is obvious how to use our techniques to get a protocol for proving that n is the product of two *strong* primes [22], i.e., $(p-1)/2, (q-1)/2, (p+1)/2$ and $(q+1)/2$ are primes or have a large prime factor. 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.

References

[1] E. Bach and J. Shallit. Factoring with cyclotomic polynomials. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 443–450, 1985.

[2] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 425–439. Springer Verlag, 1997.

[3] J. Boyar, K. Friedl, and C. Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. *Journal of Cryptology*, 4(3):185–206, 1991.

[4] S. Brands. Electronic cash systems based on the representation problem in groups of prime order. In *Preproceedings of Advances in Cryptology — CRYPTO '93*, pages 26.1–26.15, 1993.

[5] S. Brands. Rapid demonstration of linear relations connected by boolean operators. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 318–333. Springer Verlag, 1997.

[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 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.

[8] 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.

[9] A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer Verlag, 1998.

[10] D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In D. Chaum and W. L. Price, editors, *Advances in Cryptology — EUROCRYPT '87*, volume 304 of *Lecture Notes in Computer Science*, pages 127–141. Springer-Verlag, 1988.

[11] D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 200–212. Springer-Verlag, 1987.

[12] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.

[13] H. Cohen. *A Course in Computational Algebraic Number Theory*. Number 138 in Graduate Texts in Mathematics. Springer-Verlag, Berlin, 1993.

[14] R. Cramer and I. Damgård. Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *Lecture Notes in Computer Science*, pages 424–441, Berlin, 1998. Springer Verlag.

[15] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Verlag, 1994.

[16] 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.

[17] 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.

[18] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In N. Koblitz, editor, *Advances in Cryptology — CRYPT0 '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 157–172, Berlin, 1996. IACR, Springer Verlag.

[19] R. Gennaro, H. Krawczyk, and T. Rabin. RSA-based undeniable signatures. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 132–149. Springer Verlag, 1997.

[20] 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.

[21] O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer-Verlag, 1987.

[22] J. Gordon. Strong RSA keys. *Electronics Letters*, 20(12):514–516, 1984.

[23] K. Koyama, U. Maurer, T. Okamoto, and S. Vanstone. New public-key schemes based on elliptic curves over the ring Z_n. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 1992.

[24] E. Kranakis. *Primality and Cryptography*. Wiley-Teubner Series in Computer Science, 1986.

[25] D. J. Lehmann. On primality tests. *SIAM Journal of Computing*, 11(2):374–375, May 1982.

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

[27] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Verlag, 1992.

[28] J. M. Pollard. Theorems on factorization and primality testing. *Proc. Cambridge Philosophical Society*, 76:521–528, 1974.

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

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

[31] R. Solovay and V. Strassen. A fast monte-carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, Mar. 1977.

[32] J. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 128–134. Springer-Verlag, 1988.

[33] H. C. Williams. A $p+1$ method of factoring. *Mathematics of Computation*, 39(159):225–234, 1982.

A Proving a Secret's Length in Groups with Known Order

In this section we review the statistical zero-knowledge proof of knowledge of the discrete logarithm, say x, of y to the base g and that, additionally, x lies within a given interval. This proof is based on a protocols that appeared in [16, 9].

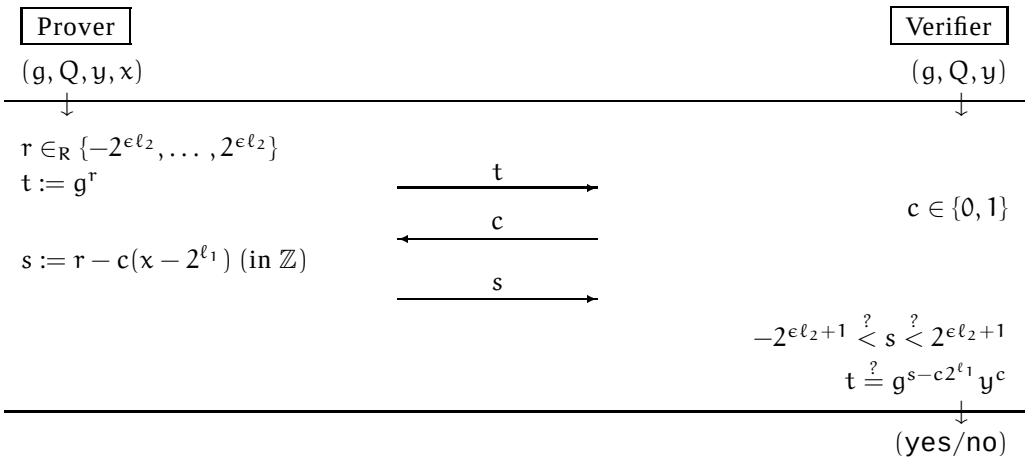

Figure 1: When repeated sufficiently many times, this protocol allows a prover to convince a verifier that the secret he committed to in y lies within the bound $\{2^{\ell_1} - 2^{\epsilon\ell_2+2}, \ldots, 2^{\ell_1} + 2^{\epsilon\ell_2+2}\}$ for given parameters ϵ, ℓ_1, and ℓ_2.

Theorem 7. *Let* G *be a group of prime order* Q, g *and* h *two generators of* G *such that* $\log_g h$ *is unknown,* $\epsilon > 1$ *be a security parameter, and* $\ell_1 < \log Q$ *and* ℓ_2 *be lengths. Let* y *be a public group element. When sequentially repeated* $j = \Theta(\mathrm{poly}(\log Q))$ *times, the*

protocol depicted in Figure 1 is a proof of knowledge of a secret $x \in \{2^{\ell_1} - 2^{\epsilon\ell_2+2}, \ldots, 2^{\ell_1} + 2^{\epsilon\ell_2+2}\}$ *such that* $y = g^x$ *and is statistical zero-knowledge for* $\ell_2 = \text{poly}(\log Q))$ *and if* $x \in \{2^{\ell_1} - 2^{\ell_2}, \ldots, 2^{\ell_1} + 2^{\ell_2}\}$.

Proof (Sketch). Proof of knowledge: First, the knowledge extractor needs two find views of accepting protocol-runs with the same t but different c's as usual. Let (t, c, s) and $(t, \tilde{c}, \tilde{s})$ these views. Without loss of generality we can assume that $c = 0$ and $\tilde{c} = 1$. Then we have $t = g^{s-2^{\ell_1}} y = g^{\tilde{s}}$ and thus $y = g^{\tilde{s}-s+2^{\ell_1}}$. Thus one can compute

$$\hat{x} := \tilde{s} - s + 2^{\ell_1} \pmod{Q}$$

such that $y = g^{\hat{x}}$ holds. Due to the bounds that s and $\tilde{s}$ satisfy, it follows that $-2^{\epsilon\ell_2+2} + 2^{\ell_1} < \hat{x} < 2^{\epsilon\ell_2+2} + 2^{\ell_1}$ must hold, which was to be shown.

Statistical zero-knowledge: We provide only a simulator for one round of the protocol. Extending this simulator to a simulator for all rounds is straight forward.

The simulator randomly chooses $c' \in_R \{0, 1\}$ and $s' \in_R \{-2^{\epsilon\ell_2}, \ldots, 2^{\epsilon\ell_2}\}$ according to the uniform distribution. Using these values, the simulator computes $t' = g^{s'} y^{c'}$ which he feeds the verifier. If the verifier responds with c' the simulator outputs (t', c', s'). To prove that these values are statistical indistinguishable from a view of a protocol run with the prover, it suffices to consider the probability distribution $P_S(s)$ of the response s of the prover and $P_{S'}(s')$ according to which the simulator chooses s'.

If the prover chooses r uniformly at random from $\{-2^{\epsilon\ell_2}, \ldots, -2^{\epsilon\ell_2}\}$ and the secret key $x - 2^{\ell_1}$ randomly from $\{-2^{\ell_2}, \ldots, 2^{\ell_2}\}$ according to any distribution, we have

$$P_S(s) \begin{cases} = 0 & \text{for } s < -2^{\epsilon\ell_2} - 2^{\ell_2} \\ \leq 2^{-\epsilon\ell_2} & \text{for } -2^{\epsilon\ell_2} - 2^{\ell_2} \leq s < -2^{\epsilon\ell_2} + 2^{\ell_2} \\ = 2^{-\epsilon\ell_2} & \text{for } -2^{\epsilon\ell_2} + 2^{\ell_2} \leq s \leq 2^{\epsilon\ell_2} - 2^{\ell_2} \\ \leq 2^{-\epsilon\ell_2} & \text{for } 2^{\epsilon\ell_2} - 2^{\ell_2} < s \leq 2^{\epsilon\ell_2} + 2^{\ell_2} \\ = 0 & \text{for } 2^{\epsilon\ell_2} + 2^{\ell_2} < s\,. \end{cases}$$

This holds for any distribution of c over $\{0, 1\}$. Thus we have

$$\sum_{\alpha \in \mathbb{Z}} |P_S(\alpha) - P_{S'}(\alpha)| \leq \frac{2^{\ell_2+2}}{2^{\epsilon\ell_2}} = \frac{4}{(2^{\ell_2})^{(\epsilon-1)}}$$

For ℓ_2 and ϵ as stated in the theorem, the last term can be expressed as one over a polynomial in the input length, and therefore the two distributions are statistical indistinguishable.

□

The relation $\epsilon\ell_2 + 2 < \log Q$ should hold, as otherwise the any secret value will lie within the bound.

In the above theorem there is a "hole" in the range of the prover's secret key x, i.e., the protocol is complete and zero-knowledge only for $(x - 2^{\ell_1}) \in \{-2^{\ell_2}, \ldots, 2^{\ell_2}\}$. For $(x - 2^{\ell_1}) \in \{-2^{\epsilon\ell_2+1}, \ldots, 2^{\epsilon\ell_2+1}\} / \{-2^{\ell_2}, \ldots, 2^{\ell_2}\}$ the prover's success probability is smaller than 1 but larger than 2^{-j}, depending on the size of $|(x - 2^{\ell_1})| - 2^{\ell_2}$.

B Square & Multiply Algorithm for $d \equiv a^b \pmod{n}$

The following algorithm computes $d \equiv a^b \pmod{n}$ (we use the same names for all variables as in Section 3.2).

$v_0 := a$
if $b_0 = 1$ **then** $u_0 := a$ **else** $u_0 := 1$ **fi**
for $i = 1$ **to** $\ell_b - 1$ **do**
  $v_i := v_{i-1} \cdot v_{i-1} \pmod{n}$ % thus $v_i \equiv a^{2^i} \pmod{n}$
  if $b_i = 1$ **then**
    $u_i := u_{i-1} \cdot v_i \pmod{n}$ % thus $u_i \equiv u_{i-1} \cdot a^{2^i} \pmod{n}$
  else
    $u_i := u_{i-1}$
  endif
endfor
$d := u_{\ell_b - 1}$

Recent BRICS Report Series Publications

RS-98-29 **Jan Camenisch and Markus Michels.** ***Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes*****. November 1998. 19 pp.**

RS-98-28 **Rasmus Pagh.** ***Low Redundancy in Dictionaries with*** $O(1)$ ***Worst Case Lookup Time*****. November 1998. 15 pp.**

RS-98-27 **Jan Camenisch and Markus Michels.** ***A Group Signature Scheme Based on an RSA-Variant.*** **November 1998. 18 pp. Preliminary version appeared in Ohta and Pei, editors,** ***Advances in Cryptology: 4th ASIACRYPT Conference on the Theory and Applications of Cryptologic Techniques*****, ASIACRYPT '98 Proceedings, LNCS 1514, 1998, pages 160–174.**

RS-98-26 **Paola Quaglia and David Walker.** ***On Encoding*** $p\pi$ ***in*** $m\pi$**. October 1998. 27 pp. Full version of paper to appear in** ***Foundations of Software Technology and Theoretical Computer Science: 18th Conference*****, FCT&TCS '98 Proceedings, LNCS, 1998.**

RS-98-25 **Devdatt P. Dubhashi.** ***Talagrand's Inequality in Hereditary Settings*****. October 1998. 22 pp.**

RS-98-24 **Devdatt P. Dubhashi.** ***Talagrand's Inequality and Locality in Distributed Computing*****. October 1998. 14 pp.**

RS-98-23 **Devdatt P. Dubhashi.** ***Martingales and Locality in Distributed Computing*****. October 1998. 19 pp.**

RS-98-22 **Gian Luca Cattani, John Power, and Glynn Winskel.** ***A Categorical Axiomatics for Bisimulation*****. September 1998. ii+21 pp. Appears in Sangiorgi and de Simone, editors,** ***Concurrency Theory: 9th International Conference*****, CONCUR '98 Proceedings, LNCS 1466, 1998, pages 581–596.**

RS-98-21 **John Power, Gian Luca Cattani, and Glynn Winskel.** ***A Representation Result for Free Cocompletions*****. September 1998. 16 pp.**

RS-98-20 **Søren Riis and Meera Sitharam.** ***Uniformly Generated Submodules of Permutation Modules*****. September 1998. 35 pp.**

RS-98-19 **Søren Riis and Meera Sitharam.** ***Generating Hard Tautologies Using Predicate Logic and the Symmetric Group*****. September 1998. 13 pp.**