Theoretical Computer Science 132 (1994) 291–317
Elsevier

The knowledge complexity of quadratic residuosity languages*

Alfredo De Santis and Giovanni Di Crescenzo

Dipartimento di Informatica ed Applicazioni, Università di Salerno, I-84081 Baronissi (SA), Italy

Guiseppe Persiano

Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138, USA, and Dipartimento di Matematica, Università di Catania, I-95125 Catania, Italy

Communicated by A. Salomaa
Received November 1992
Revised June 1993

Abstract

De Santis, A., G. Di Crescenzo and G. Persiano, The knowledge complexity of quadratic residuosity languages, Theoretical Computer Science 132 (1994) 291–317.

Noninteractive perfect zero-knowledge (ZK) proofs are very elusive objects. In fact, since the introduction of the noninteractive model of Blum et al. (1988), the only perfect zero-knowledge proof known was the one for quadratic nonresiduosity of Blum et al. (1991). The situation is no better in the interactive case where perfect zero-knowledge proofs are known only for a handful of particular languages.

In this work, we show that a large class of languages related to quadratic residuosity admits *noninteractive perfect zero-knowledge proofs*. More precisely, we give a protocol for the language of thresholds of quadratic residuosity.

Moreover, we develop a new technique for converting noninteractive zero-knowledge proofs into *round-optimal zero-knowledge* proofs for an even wider class of languages. The transformation preserves perfect zero knowledge in the sense that, if the noninteractive proof we started with is a perfect zero-knowledge proof, then we obtain a round-optimal perfect zero-knowledge proof. The noninteractive perfect zero-knowledge proofs presented in this work can be transformed into 4-round (which is optimal) interactive perfect zero-knowledge proofs. Until now, the only known 4-round perfect ZK proof systems were the ones for quadratic nonresiduosity (Goldwasser et al., 1989) and for graph nonisomorphism (Goldreich et al., 1986) and no 4-round perfect zero-knowledge proof system was known for the simple case of the language of quadratic residues.

Correspondence to: A. De Santis, Dipartimento di Informatica ed Applicazioni, Università di Salerno, I-84081 Baronissi (SA), Italy. Email: ads@udsab.dia.unisa.it.

* This work was partially supported by MURST and CNR.

SSDI 0304-3975(93)E0169-5

1. Introduction

The concept of a zero-knowledge (ZK) proof has been introduced in [25] that gave zero-knowledge proofs for the number-theoretic languages of quadratic residuosity and quadratic nonresiduosity modulo a composite integer. Neither language is believed to be in BPP. A zero-knowledge proof is a special kind of proof that allows an all-powerful prover to convince a poly-bounded verifier that a certain statement is true without revealing any additional information.

The theory of zero knowledge has been greatly extended by the work of [21] which proved that all NP languages indeed have zero-knowledge proofs. This breakthrough work caused much excitement both for its theoretical importance and for its impact on the design of cryptographic protocols [22].

The zero-knowledge proofs for all NP of [21] differs in a very substantial way from the proofs given in [25]: they are *computational* zero knowledge, i.e. secure against poly-bounded adversaries; whereas the proofs of [25] are perfect, i.e. secure against unlimited-power adversaries. Thus, the proofs of [21] are based on the unproven complexity assumption of the existence of one-way functions. Perfect zero knowledge is a desirable property for a proof as one can never be sure of the computational power of the person he is giving the proof to. On the other hand, it is very unlikely that perfect zero-knowledge proofs for all NP exist, as their complexity-theoretic consequences (the collapse of the polynomial hierarchy [10, 18]) are considered to be false. However, perfect zero-knowledge proofs have been given for some languages in NP which are not believed to be neither NP-complete nor in BPP and are either number-theoretic or have the property of random self-reducibility [11, 20, 21, 25, 29]. Because of their importance, obtaining perfect-ZK proofs for certain classes of languages still remains an important research area.

A second problem that afflicts the proofs of [21] (as well as some of the proofs of [25]) is that they require an unbounded number of rounds of communication. This severely limits the applicability of ZK proofs and has motivated the study of communication complexity of ZK proofs along two main lines of research. In [8] the possibility of disposing of interaction between prover and verifier in ZK proofs assuming that they share beforehand a short random string (see [7] for improvements, formalizations and proofs) was investigated. In this setting the only perfect ZK proof given is that for quadratic nonresiduosity modulo integers with two prime factors of [7]. Along a different line of thought, [19] investigated the round complexity of zero-knowledge proofs in the original interactive model of [25] and proved that at least 4 rounds of communication are needed to obtain nontrivial ZK proofs with black-box simulation. This result is complemented by that of [3] that gave a 5-round ZK proof for graph isomorphism. The communication complexity of ZK has been studied also for the dual model of [12]. In [13] a constant-round zero-knowledge proof for all NP has been given.

In this work we consider the problem of obtaining zero-knowledge proofs that are perfect and do not have an unbounded number of rounds of communication.

1.1 Organization of the paper and our results

In Section 2 we review some number-theoretic results about quadratic residuosity and the definition of perfect zero knowledge in the noninteractive model of [7].

In Section 3 we present two simple proof systems. The first is to prove that an integer is a Blum integer while the second is for the *logical or* of quadratic non-residuosity. More precisely, for the language OR of triples (x, y_1, y_2) such that at least one of y_1, y_2 is a quadratic nonresidue modulo x and x is a Blum integer.

In Section 4, we present our main result: a noninteractive perfect zero-knowledge proof system for any threshold gate of quadratic residuosity of any number of inputs. More precisely, for the language $T(k,m)$ of $(m+1)$-tuples $(x, y_1, \ldots, y_m)$ such that less than k of the y_i's are quadratic nonresidues modulo x and x is a Blum integer. We give a way of constructing a set of shares from the random string that has the following property. If less than k of the y_i's are quadratic nonresidues modulo x, then this set can be opened by the prover as a sharing [28] both of the bit $b=0$ and of the bit $b=1$. On the other hand, if at least k of the y_i's are quadratic nonresidues modulo x, then this set can be opened in a unique way. Then a bit b is taken from the random string and the prover has to construct a set of shares for it. Thus, if the input pair $(x, \vec{y})$ does not belong to $T(k,m)$, the prover has probability less than $1/2$ of success. By repeating m times the protocol with different pieces of the reference string, we force the probability of cheating to be negligible. The construction of the shares employs the protocol for the language OR of Section 3.

In Section 5, we show a way of obtaining a 4-round interactive zero-knowledge proof system from noninteractive ones. This result is *optimal* in view of a lower bound of [19] on the number of rounds necessary for nontrivial zero knowledge. Besides the ones for quadratic nonresiduosity [25] and for graph nonisomorphism [21], our proof systems are the only 4-round perfect zero-knowledge proof systems known. For example, the noninteractive perfect ZK proof system of Section 4 can be transformed into a 4-round interactive perfect ZK proof.

In all our proof systems, the prover's program can be performed in polynomial time provided that the factorization of the modulus is given as an additional input.

1.2. Knowledge complexity

This work has been motivated in part by the work of Goldreich and Petrank [23]. In their work, they gave definition for the concept of a proof that leaks k bits of knowledge (with zero knowledge being the case $k=0$). As far as computational zero knowledge is concerned, it is known that everything that has an interactive proof has a zero-knowledge proof [5, 26]. The question is not as clear for perfect zero knowledge. That is, are there languages in $KC(1)$ (i.e., the class of languages that can be proved releasing exactly 1 bit of knowledge) but not in $KC(0)$? Or in general, in $KC(k)$ but not in $KC(k-1)$? We know already, from the result of Fortnow [18], that NP-complete languages are not likely to be in $KC(0)$.

The most obvious candidate for such languages are languages that can be constructed from languages in $KC(0)$. However, only few languages are known to have perfect zero-knowledge proofs and these can be divided roughly into two classes: quadratic residuosity languages (or more generally number-theoretic languages) and graph-isomorphism languages.

The results of this paper prove that a large class of quadratic residuosity languages have perfect zero-knowledge proofs. In [15] essentially the same result is proved for graph-isomorphism languages. Thus, the quest for languages in $KC(1)$ but not in $KC(0)$ is still open.

2. Background and notations

2.1. Basic definitions

2.1.1. Notations

We denote by $\mathbb{N}$ the set of natural numbers. If $n \in \mathbb{N}$, by 1^n we denote the concatenation of n 1's. We identify a binary string σ with the integer x whose binary representation (with possible leading zeroes) is σ.

If σ and τ are binary strings, we denote their concatenation by either $\sigma \circ \tau$ or $\sigma\tau$.

By the expression $\vec{w}$ we denote the k-tuple $(w_1, \ldots, w_k)$ of numbers or bits. We often say that $z \in \vec{w}$, meaning that there exists i such that $z = w_i$ and $\vec{w} \in S$ meaning that $w_i \in S$, for $i = 1, \ldots, k$.

By the expression $|x|$ we denote the length of x if x is a string, the length of the binary string representing x if x is an integer, the absolute value of x if x is a real number, or the cardinality of x if x is a set. If $\vec{x}$ is a k-tuple, by the expression $|\vec{x}|$ we denote the number k of components of $|\vec{x}|$.

We use the symbol $\oplus$ to denote the bitwise xor of two binary strings of the same length. A language is a subset of $\{0,1\}^*$.

2.1.2. Models of computation

An algorithm is a Turing machine. An *efficient* algorithm is a probabilistic Turing machine running in expected polynomial time.

We emphasize the number of input received by an algorithm as follows. If algorithm A receives only one input we write $A(\cdot)$, if it receives two inputs we write $A(\cdot,\cdot)$ and so on.

2.1.3. Algorithms and probability spaces

If $A(\cdot)$ is a probabilistic algorithm, then for any input x, the notation $A(x)$ refers to the probability space that assigns to a string σ the probability that A, on input x, outputs σ.

If S is a probability space, then $x \leftarrow S$ denotes the algorithm which assigns to x an element randomly selected according to S. If F is a finite set, then the notation $x \leftarrow F$

denotes the algorithm which assigns to x an element selected according to the probability space whose sample space is F and with uniform probability distribution on the sample points.

If $p(\cdot,\cdot,\dots)$ is a predicate, the notation $\mathbf{Pr}(x\leftarrow S; y\leftarrow T; \dots : p(x,y,\dots))$ denotes the probability that $p(x,y,\dots)$ will be true after the ordered execution of the algorithms $x\leftarrow S$, $y\leftarrow T, \dots$

The notation $\{x\leftarrow S; y\leftarrow T; \dots : (x,y,\dots)\}$ denotes the probability space over $\{(x,y,\dots)\}$ generated by the ordered execution of the algorithms $x\leftarrow S$, $y\leftarrow T, \dots$

2.2. *Chernoff bounds*

The following well-known bounds on the tails of the binomial distribution (see, for instance, [2, 16]) will be used in proving the completeness and soundness of our proof systems.

Fact 2.1. *Let $S_{n,p}$ be the probability space whose distribution is binomial with parameters n, p, and let ε be a constant in the range $0\leqslant\varepsilon<1$. Then*

$$\mathbf{Pr}(X\leftarrow S_{n,p}: X\geqslant(1+\varepsilon)np)\leqslant\exp(-\varepsilon^2 np/2),$$

$$\mathbf{Pr}(X\leftarrow S_{n,p}: X\leqslant(1-\varepsilon)np)\leqslant\exp(-\varepsilon^2 np/3).$$

2.3. *Number theory*

2.3.1. *Quadratic residuosity*

For each integer $x>0$, the set of integers less than x and relatively prime to x form a group under multiplication modulo x denoted by Z_x^*. We say that $y\in Z_x^*$ is a *quadratic residue* modulo x iff there is a $w\in Z_x^*$ such that $w^2\equiv y \bmod x$. If this is not the case, we call y a *quadratic nonresidue* modulo x. For compactness, we define the *quadratic residuosity predicate* as follows:

$$\mathcal{Q}_x(y)=\begin{cases}0 & \text{if } y \text{ is a quadratic residue modulo } x,\\ 1 & \text{otherwise.}\end{cases}$$

Fact 2.2 (see, for instance, Niven and Zuckerman [27]). *If $y_1, y_2\in Z_x^*$, then*

(1) $\mathcal{Q}_x(y_1)=\mathcal{Q}_x(y_2)=0 \Rightarrow \mathcal{Q}_x(y_1y_2)=0$,

(2) $\mathcal{Q}(y_1)\neq\mathcal{Q}_x(y_2) \Rightarrow \mathcal{Q}_x(y_1y_2)=1$.

The quadratic residuosity predicate defines the following equivalence relation in Z_x^*: $y_1\sim_x y_2$ if and only if $\mathcal{Q}_x(y_1y_2)=0$. Thus, the quadratic residues modulo x form a $\sim_x$ equivalence class. More generally, it is immediately seen that the following fact is true.

Fact 2.3. *For any fixed $y \in Z_x^*$, the elements $\{yq \bmod x \mid q$ is a quadratic residue modulo $x\}$ constitute a $\sim_x$ equivalence class that has the same cardinality as the class of quadratic residues.*

For p prime, the problem of deciding quadratic residuosity coincides with the problem of computing the Legendre symbol. In fact, for p prime and $y \in Z_p^*$, the Legendre symbol $(y \mid p)$ of y modulo p is defined as follows:

$$(y \mid p) = \begin{cases} +1 & \text{if } y \text{ is a quadratic residue modulo } x, \\ -1 & \text{otherwise,} \end{cases}$$

and can be computed in polynomial time by using Euler's criterion. Namely, $(y \mid p) = y^{(p-1)/2} \bmod p$.

Euler's criterion and the following fact give an efficient algorithm for deciding quadratic residuosity modulo integer whose factorization is known.

Fact 2.4 (see, for instance, Niven and Zuckerman [27]). *y is a quadratic residue modulo x if and only if y is a quadratic residue modulo each of the prime divisors of x.*

However, no efficient algorithm is known for deciding quadratic residuosity modulo composite numbers whose factorization is not given. Some help is provided by the Jacobi symbol which extends the Legendre symbol to composite integers as follows

$$(y \mid x) = \prod_{i=1}^{n} (y \mid p_i)^k,$$

where $x = p_1^{k_1} \cdots p_n^{k_n}$ and the k_i's are positive integers and the p_i's are distinct primes.

Despite the fact that the Jacobi symbol is defined in terms of the factorization of the modulus, we have the following fact.

Fact 2.5 (see Angluin [1] or Niven and Zuckerman [27]). *The Jacobi symbol can be computed in deterministic polynomial time.*

Define Z_x^{+1} and Z_x^{-1} to be, respectively, the sets of elements of Z_x^* with Jacobi symbol $+1$ and -1 and $QR_x = \{y \in Z_x^* \mid \mathcal{Q}_x(y) = 0\}$, $NQR_x = \{y \in Z_x^* \mid (y \mid x) = +1, \mathcal{Q}_x(y) = 1\}$.

It can be immediately seen that if $y \in Z_x^{-1}$, then it is not a quadratic residue modulo x, as it is not a quadratic residue modulo some prime p_i dividing x. However, if $y \in Z_x^{+1}$, no efficient algorithm is known to compute $\mathcal{Q}_x(y)$. The fastest way known for computing $\mathcal{Q}_x(y)$ consists of first factoring x. This fact has been first used in Cryptography by Goldwasser and Micali [24].

2.3.2. Blum integers

In this paper we will be mainly concerned with the special moduli called Blum integers.

Definition 2.6. An integer x is a Blum integer, in symbols $x \in BL$, if and only if $x = p^{k_1} q^{k_2}$, where p and q are different primes both $\equiv 3 \bmod 4$ and k_1 and k_2 are odd integers.

It follows from Fact 2.4 and Euler's criterion that if x is a Blum integer, $-1 \bmod x$ is a quadratic nonresidue with Jacobi symbol $+1$. Moreover we have the following fact.

Fact 2.7. *On input a Blum integer x, it is easy to generate a random quadratic nonresidue in Z_x^{+1}: randomly select $r \in Z_x^*$ and output $-r^2 \bmod x$.*

The following lemmas prove that the Blum integers enjoy the elegant property that each quadratic residue has a square root which is itself a quadratic residue. Thus each quadratic residue modulo a Blum integer has also a fourth root.

Lemma 2.8. *Let p be a prime $\equiv 3 \bmod 4$. The mapping $x \to x^2 \bmod p$ is $1-1$ over QR_p, and every quadratic residue has a unique square root which is also a quadratic residue modulo p.*

Proof. Let x_p and $-x_p$ be the two square roots of $x \bmod p$. By Euler's criterion, $(-1 \mid p) = -1$, and thus $-1 \bmod x$ is a quadratic nonresidue modulo p. So, from Fact 2.2, x_p and $-x_p$ have different quadratic residuosity. Thus, if x_p is a quadratic nonresidue, then $-x_p$ must be a quadratic residue and vice versa. □

Lemma 2.9. *Let x be a Blum integer. Every quadratic residue modulo x has at least one square root which is itself a quadratic residue modulo x.*

Proof. Let $x \in BL$ and y a quadratic residue modulo x. Then, by Fact 2.4, y is a quadratic residue modulo the primes p, q dividing x. We call y_p, $-y_p$ and y_q, $-y_q$ the square roots of y, respectively, modulo p and q such that y_p and y_q are quadratic residues (we know of their existence by Lemma 2.8). Moreover, by the Chinese remainder theorem, there exists a z such that $z \equiv y_p \bmod p$ and $z \equiv y_q \bmod q$. Also, $z \in Z_x^*$ is a quadratic residue and a square root of $y \bmod x$. □

Similar to the previous lemma one can prove the following lemma.

Lemma 2.10. *Let $x = p^{k_1} q^{k_2}$, where $p \equiv 1 \bmod 4$. Then, at least one half of the quadratic residues have no square root which is itself a quadratic residue modulo x.*

2.3.3. Regular integers

Following [7], we define an integer x to be *regular* if it enjoys the elegant structural property $|Z_x^{+1}| = |Z_x^{-1}|$. We define *Regular*(s) to be the set of regular integers with s distinct prime divisors. By the definition of Jacobi symbol, the following fact is straightforward.

Fact 2.11. *An odd integer x belongs to Regular(s) if and only if it has s distinct prime factors and is not a perfect square.*

Equivalently, by Fact 2.3, we have the following.

Fact 2.12. *An odd integer x belongs to Regular(s) if and only if it is regular and Z_x^* is partitioned by $\sim_x$ into 2^s equally numerous equivalence classes. (Equivalently, Z_x^{+1} is partitioned by $\sim_x$ into 2^{s-1} equally numerous equivalence classes.*

Therefore, if x is an odd integer belonging to *Regular*(2), Z_x^{+1} is partitioned by $\sim_x$ into 2 equally numerous equivalence classes, one made of quadratic residues modulo x and the other made of quadratic nonresidues modulo x. Thus, for this special class of integers we have the following fact.

Fact. 2.13. *Let x be an odd integer belonging to Regular*(2). *If $y_1, y_2 \in Z_x^*$, then*

(1) $\mathcal{Q}_x(y_1) = \mathcal{Q}_x(y_2) \Rightarrow \mathcal{Q}_x(y_1 y_2) = 0$,

(2) $\mathcal{Q}_x(y_1) \neq \mathcal{Q}_x(y_2) \Rightarrow \mathcal{Q}_x(y_1 y_2) = 1$.

From Lemmas 2.2 and 2.3, one can prove the following fact that completely characterizes Blum integers with respect to *Regular* (2) integers.

Fact 2.14. *An integer x is a Blum integer if and only if*

(1) $x \in$ *Regular* (2),

(2) $-1 \bmod x \in NQR_x$,

(3) *for each $w \in QR_x$ there exists an r such that $r^4 \equiv w \bmod x$.*

2.4. Non-interactive perfect zero knowledge

The shared-string model for noninteractive ZK has been put forward in [8] and further elaborated by [7] (see also [14]). In this model, prover and verifier share a random string and the communication is monodirectional. In [7] it is proved that under the quadratic residuosity assumption all NP languages have non-interactive computational zero-knowledge proofs in this model. Also, they gave a perfect noninteractive ZK proof for the language of quadratic nonresiduosity modulo *Regular* (2) integers. Subsequently, in [17] it was proved that certified trapdoor permutations are sufficient for proving noninteractively and with zero knowledge the membership to any language in NP ([4] showed how a trapdoor permutation can be certified in a noninteractive fashion).

Let us now review the definition of noninteractive perfect ZK of [7] (We refer the reader to the original paper for motivations and discussion of the definition.)

We denote by L the language in question and by x an instance to it. Let c be a positive constant, P a probabilistic Turing machine and V a deterministic Turing machine that runs in time polynomial in the length of its first input.

Definition 2.15. We say that (P, V) is a Noninteractive perfect zero-knowledge proof system (noninteractive perfect ZK proof system) for the language L if there exists a positive constant c such that:

(1) *Completeness.* $\forall x \in L$, $|x|=n$ and for all sufficiently large n,

$$\mathbf{Pr}(\sigma \leftarrow \{0,1\}^{n^c}; \mathit{Proof} \leftarrow P(\sigma, x): V(\sigma, x, \mathit{proof})=1) > 1-2^{-n}.$$

(2) *Soundness.* For all probabilistic algorithms *Adversary* outputting pairs (x, Proof), where $x \notin L$, $|x|=n$, and all sufficiently large n,

$$\mathbf{Pr}(\sigma \leftarrow \{0,1\}^{n^c}; (x, \mathit{Proof}) \leftarrow \mathit{Adversary}(\sigma): V(\sigma, x, \mathit{Proof})=1) < 2^{-n}.$$

(3) *Perfect zero knowledge.* There exists an efficient simulator algorithm S such that $\forall x \in L$, $|x|=n$, the two probability spaces $S(x)$ and $\mathit{View}_V(x)$ are equal, where by $\mathit{View}_V(x)$ we denote the probability space

$$\mathit{View}_V(x) = \{\sigma \leftarrow \{0,1\}^{n^c}; \mathit{Proof} \leftarrow P(\sigma, x): (\sigma, \mathit{Proof})\}.$$

We note that in soundness, we let the adversary choose the false statement he wants to prove after seeing the random string. Nonetheless, he has only negligible probability of convincing V.

We say that (P, V) is a noninteractive proof system for the language L if completeness and soundness are satisfied.

We call the "common" random string σ, input to both P and V, the *reference string.* (Above, the common input is σ and x.)

3. Non-interactive perfect zero knowledge for BL and OR

In this section we discuss a simple proof system for the language BL of Blum integers. Then we give a proof system for the language OR of logical or of quadratic residuosity that we will define later.

A proof system (A, B) for BL is easily obtained using the characterization of Blum integers given by Fact 2.14. In fact, it is sufficient for the prover to first prove that x is a *Regular* (2) integer and that -1 is a quadratic nonresidue using the proof system given in [7]. Then, all it is left to prove is that every quadratic residue has a fourth root modulo x. This is done by giving, for each element $y \in Z_x^{+1}$ taken from the random string, a fourth root modulo x of y or $-y$, depending on the quadratic residuosity of y. Completeness, soundness, and perfect zero knowledge are easily seen to be satisfied.

We now give a noninteractive perfect ZK proof system (C, D) for the language

$$\mathrm{OR} = \{(x, y_1, y_2) \mid x \in \mathrm{BL},\ y_1, y_2 \in Z_x^{+1} \quad \text{and} \quad (y_1 \in NQR_x) \vee (y_2 \in NQR_x)\}.$$

This is an extension of the proof system for quadratic nonresiduosity given in [7].

The prover C wants to convince the polynomial-time verifier that at least one of the two integers y_1, y_2 is a quadratic nonresidue modulo a Blum integer x without giving away any information that D was not able to compute alone before. D cannot compute by himself if $(x, y_1, y_2) \in \mathrm{OR}$, because the fastest way known for deciding quadratic residuosity modulo a composite integer x consists of factoring x, thus also for this problem no efficient algorithm is known. Moreover, the proof is noninteractive (C gives only one message to D) and perfect zero knowledge (D does not gain any additional information even if not restricted to run in polynomial time).

In our construction we will use the following definition.

Definition 3.1. For any positive integer x, define the relation $\approx_x$ on $Z_x^{+1} \times Z_x^{+1}$ as follows:

$$(a_1, a_2) \approx_x (b_1, b_2) \Leftrightarrow a_1 \sim_x b_1 \text{ and } a_2 \sim_x b_2.$$

We write $(a_1, a_2) \not\approx_x (b_1, b_2)$ when (a_1, a_2) is not $\approx_x$ equivalent to (b_1, b_2). From Fact 2.3, one can prove that for each integer $x \in Regular(s)$, $\approx_x$ is an equivalence relation on $Z_x^{+1} \times Z_x^{+1}$ and that there are $2^{2(s-1)}$ equally numerous $\approx_x$ equivalence classes.

3.1.1. An informal description

Let us informally describe the protocol (C, D). A formal description of (C, D) can be found in Figs. 1 and 2. By (A, B) we denote the perfect noninteractive zero-knowledge proof system for the language BL described above. On input (x, y_1, y_2) with $|x| = n$, C and D share a string γ of length $660n^2$. This string is split into $\rho \circ \sigma_{1,1} \circ \sigma_{1,2} \circ \cdots \circ \sigma_{300n,1} \circ \sigma_{300n,2}$, where ρ has length $60n^2$ and each $\sigma_{i,j}$ has length n. First C proves that $x \in \mathrm{BL}$ by running the algorithm A on input x and using the random string ρ. C partitions the pairs $(\sigma_{i,1}, \sigma_{i,2})$ belonging to $Z_x^{+1} \times Z_x^{+1}$ according to the relation $\approx_x$ into 4 classes. It is easy for C to prove that two pairs $(\sigma_{i,1}, \sigma_{i,2})$ and $(\sigma_{j,1}, \sigma_{j,2})$ belong to the same class: C just gives a square root modulo x of the products $\sigma_{i,1}\sigma_{j,1} \bmod x$ and $\sigma_{i,2}\sigma_{j,2} \bmod x$. Once all the pairs, including the input pair (y_1, y_2), have been assigned to an equivalence class, C uncovers the class of pairs made of two quadratic residues by giving the square root of both elements of one of its pairs. D checks that the pair (y_1, y_2) is in a different class from that whose pairs are both quadratic residues.

Now, suppose $(x, y_1, y_2) \notin \mathrm{OR}$. Then C can perform the protocol if and only if one of the three classes of pairs, for which at least one element is a quadratic nonresidue, does not appear in the random string. In fact the prover has to uncover the class of pairs made of two quadratic residues and thus (y_1, y_2) has to be assigned to one of the three

Input to C and D:

- $(x, y_1, y_2) \in \mathrm{OR}$, $|x| = n$.
- A $660n^2$-bit random string γ.

  (Set $\gamma = \tau \circ \sigma_{1,1} \circ \sigma_{1,2} \circ \cdots \circ \sigma_{300n,1} \circ \sigma_{300n,2}$, where τ has length $60n^2$ and each $\sigma_{i,j}$ has length n).

Instructions for C.

C.1. Set Proof=empty string.

C.2. (*Prove that x is a Blum integer.*)
Run A's algorithm on input x using the random string τ and obtaining as output a string *Pf*. Append *Pf* to Proof.

C.3. (*Form the pairs* $(\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4)$.)

C.3.1. Set $\alpha_1 \leftarrow y_1$ and $\beta_1 \leftarrow y_2$.

Choose at random 3 pairs (α_2, β_2), (α_3, β_3), (α_4, β_4) in $Z_x^{+1} \times Z_x^{+1}$ such that

(a) $(\alpha_i, \beta_i) \not\approx_x (\alpha_j, \beta_j)$ for $1 \leqslant i < j \leqslant 4$, and

(b) $\mathcal{Q}_x(\alpha_2) = \mathcal{Q}_x(\beta_2) = 0$;

append $(\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4)$ to Proof.

C.3.2. Randomly choose (a, b), such that $a^2 \equiv \alpha_2 \bmod x$ and $b^2 \equiv \beta_2 \bmod x$, and append (a, b) to Proof.

C.4. (*Divide the pairs from the reference string into the four* $\approx_x$ *equivalence classes.*)

For $i = 1, \ldots, 300n$.

if $(\sigma_{i,1}, \sigma_{i,2}) \in Z_x^{+1} \times Z_x^{+1}$ then

choose j_i, $1 \leqslant j_i \leqslant 4$, such that $(\sigma_{i,1}, \sigma_{i,2}) \approx_x (\alpha_{j_i}, \beta_{j_i})$;

randomly choose $(s_i, t_i) \in Z_x^* \times Z_x^*$ such that $s_i^2 \equiv \alpha_{j_i} \sigma_{i,1} \bmod x$ and $t_i^2 \equiv \beta_{j_i} \sigma_{i,2} \bmod x$;

else set $j_i \leftarrow 0$, $s_i \leftarrow 0$, $t_i \leftarrow 0$;

append (i, j_i, s_i, t_i) to Proof.

C.5. Send Proof.

Fig. 1. The prover C for OR.

remaining classes. However, this means that all the pairs in that class must be made of two quadratic residues and thus we would only have representatives from three classes. This happens with negligible probability.

Theorem 3.2. (C, D) *is a noninteractive proof system for the language* OR.

Proof. D runs in polynomial time. In fact, B runs in polynomial time, the Jacobi symbol can be computed in polynomial time and the other steps are trivial.

Completeness: Assume $(x, y_1, y_2) \in \mathrm{OR}$. Then step D.2 is passed with high probability because of the completeness of (A, B). Steps D.5 is always passed.

Input to D:

- The string Proof sent by C.

Instructions for D.

D.1. Let Proof be the sequence
$(Pf, (\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4), (a, b)\,(1, j_1, s_1, t_1), \ldots, (300n, j_{300n}, s_{300n}, t_{300n}))$.

D.2. (*Verify that x is a Blum integer.*)
Run B's algorithm on input x, τ and Pf.

D.3. If $y_1 \in Z_x^{-1}$ or $y_2 \in Z_x^{-1}$ then HALT and REJECT.

D.4. If $(\sigma_{i,1}, \sigma_{i,2}) \in Z_x^{+1} \times Z_x^{+1}$ for less than $4n$ indices i then HALT and ACCEPT.

D.5. (*Check that the pair from the reference string have been divided into four* $\approx_x$ *equivalence classes.*)

D.5.1. Verify that $a^2 \equiv \alpha_2 \bmod x$ and $b^2 \equiv \beta_2 \bmod x$.

D.5.2. For $i = 1, \ldots, 300n$,
if $(\sigma_{i,1}, \sigma_{i,2}) \in Z_x^{+1} \times Z_x^{+1}$ then
verify that $s_i^2 \equiv \alpha_{j_i} \sigma_{i,1} \bmod x$ and $t_i^2 \equiv \beta_{j_i} \sigma_{i,2} \bmod x$.

If all verifications are successful then ACCEPT else REJECT.

Fig. 2. The verifier D or OR.

Soundness: First note that D halts at step D.4 with negligible probability regardless of whether (x, y_1, y_2) belongs to OR or not. In fact, a random n-bit integer belongs to Z_x^{+1} with probability at least 1/8. Thus the probability that $\sigma_{i,1} \notin Z_x^{+1}$ or $\sigma_{i,2} \notin Z_x^{+1}$ for less than $4n$ indices i is, by Chernoff bound, at most e^{-n}. Therefore, the probability that there exists an n-bit modulus x for which this happens is at most $2^n e^{-n}$, which is negligible.

Suppose $(x, y_1, y_2) \notin$ OR. Then we have three cases:

(a) $x \notin$ BL;

(b) $x \in$ BL, but $(y_1, y_2) \notin Z_x^{+1} \times Z_x^{+1}$.

(c) $x \in$ BL, but $y_1, y_2 \in QR_x$.

If case (a) occurs, D halts at step D.2 with overwhelming probability because of the soundness of (A, B). Then, if case (b) occurs, D halts at step D.3 with probability 1.

Let us now examine case (c). For verification step D.5.1 to be passed, C must exhibit a square root of α_2 and β_2. Then, for verification steps D.5.2 to be passed, C must partition the pairs of elements of the random string belonging to Z_x^{+1} into 4 equivalence classes with respect to the $\approx_x$ relation. We note that the first class is made of pairs $(\sigma_{i,1}, \sigma_{i,2}) \approx_x (y_1, y_2)$ and the second of pairs $(\sigma_{i,1}\, \sigma_{i,2}) \approx_x (\alpha_2, \beta_2)$, i.e. both classes are made of pairs of quadratic residues. Thus, all pairs of elements $(\sigma_{i,1}, \sigma_{i,2})$ in $Z_x^{+1} \times Z_x^{+1}$ must belong to the union of at most 3 $\approx_x$ equivalence classes, one of which is made of pairs of quadratic residues. But the probability of this event is less than $3(3/4)^{4n}$; this can be explained as follows: 3/4 is the probability that each pair belongs to the union of 3 fixed equivalence classes, there are at least $4n$ pairs, there are at most $\binom{3}{2} = 3$ ways to choose 2 classes out of 3 (note that the class of pairs made of quadratic

residues must be one of these three classes). Therefore, the probability that there exists an n-bit integer x such that this event occurs is at most $2^n 3(3/4)^{4n}$, which is negligible. □

To prove the perfect zero-knowledge property, we show an efficient simulator F such that, for each $(x, y_1, y_2) \in \mathrm{OR}$, the probability space $\mathrm{F}(x, y_1, y_2)$ is equal to $View_D(x, y_1, y_2)$ (Here M is the simulator of the perfect noninteractive ZK proof system (A, B)). A formal description of F can be found in Fig. 3.

Lemma 3.3. *The simulator* F *runs in probabilistic polynomial time and the probability spaces* $\mathrm{F}(x, y_1, y_2)$ *and* $View_D(x, y_1, y_2)$ *are equal for each* $(x, y_1, y_2) \in \mathrm{OR}$.

Proof. It is easy to see that the simulator runs in probabilistic polynomial time.

Now we just have to prove that the probability space given by the output of F and the probability space representing the view of D in the protocol are the same. First, we see that τ is equally distributed both in the protocol and in the simulator because (A, B) is a perfect noninteractive ZK proof system.

Let us see now that the pairs $(\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4)$ have been correctly constructed. It is easy to see that α_2 and β_2 are two quadratic residues. Let us now show that the pairs $(\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4)$ are pairwise $\not\approx_x$ not equivalent. We analyze the case in which the outcome b_1 of the first coin tossed by F is equal to 1 (the case $b_1 = 0$ is similar). First $(\alpha_1, \beta_1) \not\approx_x (\alpha_2, \beta_2)$ because $(x, y_1, y_2) \in \mathrm{OR}$. Next observe that $(\alpha_1, \beta_1) \not\approx_x (\alpha_3, \beta_3)$. If this were not the case, then $\mathcal{Q}_x(y_1) = \mathcal{Q}_x(\alpha_1) = \mathcal{Q}_x(\alpha_3) = \mathcal{Q}_x(y_1 y_2)$, and $\mathcal{Q}_x(y_2) = \mathcal{Q}_x(\beta_1) = \mathcal{Q}_x(\beta_3) = \mathcal{Q}_x(y_1)$, which, by Fact 2.13, implies that y_1 and y_2 are two quadratic residues, contradicting the fact that $(x, y_1, y_2) \in \mathrm{OR}$. Similarly, $(\alpha_1, \beta_1) \not\approx_x (\alpha_4, \beta_4)$. Moreover $(\alpha_2, \beta_2) \not\approx_x (\alpha_3, \beta_3)$. If this were not the case, then $\mathcal{Q}_x(y_1 y_2) = \mathcal{Q}_x(\alpha_3) = \mathcal{Q}_x(\beta_3) = \mathcal{Q}_x(y_1) = 0$, that contradicts $(x, y_1, y_2) \in \mathrm{OR}$. Similarly $(\alpha_2, \beta_2) \not\approx_x (\alpha_4, \beta_4)$. Finally $(\alpha_2, \beta_3) \not\approx_x (\alpha_4, \beta_4)$. If this were not the case, then $\mathcal{Q}_x(y_1 y_2) = \mathcal{Q}_x(\alpha_3) = \mathcal{Q}_x(\alpha_4) = \mathcal{Q}_x(y_2)$, and $\mathcal{Q}_x(y_1) = \mathcal{Q}_x(\beta_3) = \mathcal{Q}_x(\beta_4) = \mathcal{Q}_x(y_1 y_2)$, that by Fact 2.13, implies that y_1 and y_2 are two quadratic residues, and contradicts $(x, y_1, y_2) \in \mathrm{OR}$. Moreover note that, after choosing (α_1, β_1) and (α_2, β_2), F selects the other two pairs (α_3, β_3) and (α_4, β_4) randomly between the remaining two $\approx_z$ equivalence classes. To this purpose he uses the outcome of the fair coin b_1.

All that is left to prove now is that the remaining part of γ is uniformly distributed over all the binary strings of the appropriate length, or, equivalently that each pair $(\sigma_{i,1}, \sigma_{i,2})$ is uniformly distributed over the pairs of n-bit integers. If at least one of $\sigma_{i,1}, \sigma_{i,2}$ is not in Z_x^{+1}, then the pair is clearly uniformly distributed (see step 5.2). If both are in Z_x^{+1}, then $(\sigma_{i,1}, \sigma_{i,2})$ is a pair which has probability 1/4 of belonging to one of the four $\approx_x$ equivalence classes and thus is uniformly distributed (see step 5.3). Moreover, in this case it is easy that s_i and t_i are random square roots modulo x of $\alpha_j, \sigma_{i,1}$ and $\beta_{j_i} \sigma_{i,2}$. □

The above proves the following theorem.

Instructions for F.

Input: $(x, y_1, y_2) \in \mathrm{OR}$, where $|x| = n$.

1. Set Proof = empty string.
2. Run M's algorithm on input x obtaining as output (τ, Pf); append Pf to Proof.
3. If $y_1 \in Z_x^{-1}$ or $y_2 \in Z_x^{-1}$ then output (τ, Proof) and HALT.
4. (*Form the pairs* $(\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4)$.
   Randomly choose $r_1, r_2, r_3, r_4, r_5, r_6 \in Z_x^*$;
   set $\alpha_1 \leftarrow y_1$, $\beta_1 \leftarrow y_2$, $\alpha_2 \leftarrow r_1^2 \bmod x$ and $\beta_2 \leftarrow r_2^2 \bmod x$;
   set $\alpha_3' \leftarrow y_1 y_2 r_3^2 \bmod x$, $\beta_3' \leftarrow y_1 r_4^2 \bmod x$;
   set $\alpha_4' \leftarrow y_2 r_5^2 \bmod x$ and $\beta_4' \leftarrow y_1 y_2 r_6^2 \bmod x$;
   toss a fair coin and let $b_1 \in \{0, 1\}$ be its outcome;
   if $b_1 = 1$ then set $\alpha_3 \leftarrow \alpha_3', \alpha_4 \leftarrow \alpha_4', \beta_3 \leftarrow \beta_3', \beta_4 \leftarrow \beta_4'$;
   if $b_1 = 0$ then set $\alpha_3 \leftarrow \alpha_4', \alpha_4 \leftarrow \alpha_3', \beta_3 \leftarrow \beta_4', \beta_4 \leftarrow \beta_3'$;
   append $(\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4), (r_1, r_2)$ to Proof.
5. (*Distribute the pairs from the reference string into four* $\approx_x$ *equivalence classes.*)
   For $i = 1$ to $300n$,
   5.1. randomly choose two n-bit integers u_i, v_i;
   5.2. if $u_i \notin Z_x^{+1}$ or $v_i \notin Z_x^{+1}$ then
   set $\sigma_{i,1} \leftarrow u_i$, $\sigma_{i,2} \leftarrow v_i$;
   append $(i, 0, 0, 0)$ to Proof;
   5.3. if $u_i \in Z_x^{+1}$ and $v_i \in Z_x^{+1}$ then
   randomly choose two integers $s_i, t_i \in Z_x^*$;
   toss two fair coins and let $b_2, b_3 \in \{0, 1\}$ be their outcome;
   if ($b_2 = 1$ and $b_3 = 1$) then
   set $\sigma_{i,1} = y_1^{-1} s_i^2 \bmod x$, $\sigma_{i,2} = y_2^{-1} t_i^2 \bmod x$;
   append $(i, 1, s_i, t_i)$ to Proof;
   if ($b_2 = 1$ and $b_3 = 0$) then
   set $\sigma_{i,1} = s_i^2 \bmod x$, $\sigma_{i,2} = t_i^2 \bmod x$;
   append $(i, 2, r_1 s_i \bmod x, r_2 t_i \bmod x)$ to Proof;
   if (($b_1 = 0$ and $b_2 = 0$ and $b_3 = 1$) or ($b_1 = 1$ and $b_2 = 0$ and $b_3 = 0$)) then
   set $\sigma_{i,1} = y_1^{-1} y_2^{-1} s_i^2 \bmod x$, $\sigma_{i,2} = y_1^{-1} t_i^2 \bmod x$;
   append $(i, 4 - b_1, r_3 s_i \bmod x, r_4 t_i \bmod x)$ to Proof;
   if (($b_1 = 1$ and $b_2 = 0$ and $b_3 = 1$) or $b_1 = 0$ and $b_2 = 0$ and $b_3 = 0$)) then
   set $\sigma_{i,1} = y_2^{-1} s_i^2 \bmod x$, $\sigma_{i,2} = y_1^{-1} y_2^{-1} t_i^2 \bmod x$;
   append $(i, 3 + b_1, r_5 s_i \bmod x, r_6 t_i \bmod x)$ to Proof.
6. Set $\gamma = \tau \circ \sigma_{1,1} \circ \sigma_{1,2} \circ \cdots \circ \sigma_{300n,1} \circ \sigma_{300n,2}$ and **Output:** (γ, Proof).

Fig. 3. The simulator F.

Theorem 3.4. (C, D) *is a noninteractive perfect zero-knowledge proof system for the language* OR.

Remark 3.5. Using the properties of Blum integers, and in particular the fact that -1 is a quadratic nonresidue modulo x, it is possible to construct a perfect noninteractive ZK proof system for the language of triples (x, y_1, y_2) where x is a Blum integer and at least one of y_1, y_2 is a quadratic residue mod x. Just run (C, D) on input $(x, -y_1 \bmod x, -y_2 \bmod x)$.

4. Noninteractive perfect zero knowledge for threshold gates

In this section we give a noninteractive perfect zero-knowledge proof system (P, V) for the language $T(k, m)$ of pairs $(x, \vec{y})$ where less than k elements of $\vec{y} = (y_1, \ldots, y_m)$ are quadratic nonresidue modulo x. That is, the language

$$T(k, m) = \{(x, \vec{y}) \mid x \in BL,\ y_i \in Z_x^{+1},\ i = 1, \ldots, m \text{ and } |\{y_i \mid y_i \in NQR_x\}| < k\},$$

for $1 \leqslant k \leqslant m$. For instance, $T(1, m)$ is the language of pairs $(x, \vec{y})$ that satisfy $(y_1 \in QR_x) \wedge \cdots \wedge (y_m \in QR_x)$ and $T(m, m)$ is the language of pairs $(x, \vec{y})$ that satisfy $(y_1 \in QR_x) \vee \cdots \vee (y_m \in QR_x)$.

The prover P wants to convince the polynomial-time verifier V that less than k of the m integers $y_1, \ldots, y_m$ are quadratic nonresidue modulo the Blum integer x without giving away any information that V was not able to compute alone before. V cannot compute by himself whether $(x, \vec{y}) \in T(k, m)$, since the fastest way known for deciding quadratic residuosity modulo a composite integer x consists of first factoring x. Thus no efficient algorithm is known to decide if $(x, \vec{y}) \in T(k, m)$. Moreover, the proof is noninteractive (P sends only one message to V), and perfect zero-knowledge (V does not gain any additional information even if not restricted to run in polynomial time).

We use the proof systems (A, B) and (C, D) of previous sections as subroutines for (P, V).

4.1. The proof system (P, V) *for* $T(k, m)$

Before presenting our proof system for $T(k, m)$, we review the notion of *threshold scheme*, introduced by Shamir [28] and Blackley [6], that will be instrumental for our construction. A (k, m)-threshold scheme is an efficient algorithm that on input a data S outputs m *pieces* $S_1, \ldots, S_m$, such that:

- knowledge of any k or more pieces S_i makes S easily computable,
- knowledge of any $k-1$ or fewer pieces S_i leaves S completely undetermined (all its possible values are equally likely).

Shamir [28] shows how to construct such threshold schemes using interpolation of polynomials. We have the following fact.

Fact 4.1. *The following is a* (k, m)*-threshold scheme. Let* $(\mathcal{F}, +, \cdot)$ *be a finite field with more than* m *elements and let* S *be the value to be shared. Choose at random*

$a_1, \ldots, a_{k-1} \in \mathscr{E}$, *construct the polynomial* $q(x) = S + a_1 \cdot x + \cdots + a_{k-1} \cdot x^{k-1}$ *and output* $S_i = q(i)$ (*all operations are performed in* $\mathscr{E}$).

We say that a sequence $(S_1, \ldots, S_m)$ is a (k, m)-*sequence of admissible shares for* S (we will call it *sequence of admissible shares* when k and m are clear from the context) if there exists a polynomial $q(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}$ with coefficients in $\mathscr{E}$, such that $a_0 = S$ and $S_i = q(i)$ for $i = 1, \ldots, m$.

4.1.1. Observation

Let $I \subseteq \{1, \ldots, m\}$ and suppose $|I| < k$. Then given S and a sequence $(S_i | i \in I)$ of values, it is always possible to efficiently generate random values S_i, $i \notin I$, such that $(S_1, \ldots, S_m)$ is a sequence of admissible shares for S (for random values S_i, $i \notin I$ we mean that the S_i's for $i \in I$ are uniformly distributed among the S_i's such that $(S_1, \ldots, S_m)$ is a sequence of admissible shares for S). Moreover, given a sequence $(S_i | i \in I)$ of values, if the values S_i for $i \notin I$ are chosen with uniform distribution among the S_i's such that $(S_1, \ldots, S_m)$ is a sequence of admissible shares for S, then S is uniformly distributed in $\mathscr{E}$. On the other hand, if $|I| \geq k$, then a sequence $(S_i | i \in I)$ of values uniquely determines a value S and values S_i for $i \notin I$ such that $(S_1, \ldots, S_m)$ is a sequence of admissible shares for S.

Let us now introduce a bit of notation that we will use in the description of our proof system. Let $x \in \mathrm{BL}$, w and $y \in Z_x^{+1}$ and $b \in \{0, 1\}$. We define the predicate $\mathscr{B}(x, y, w, b)$ in the following way:

$$\mathscr{B}(x, y, w, b) = ((-1)^b w \bmod x \in QR_x) \vee (y \in QR_x).$$

We say that the prover (x, y)-*opens* w as b if he proves that $\mathscr{B}(x, y, w, b) = 1$.

If $y \in QR_x$ then $\mathscr{B}(x, y, w, 0) = \mathscr{B}(x, y, w, 1) = 1$ regardless of the quadratic residuosity of w and thus the prover can (x, y)-open w both as a 0 and as a 1.

Instead, if $y \in NQR_x$ then the prover can open w in just one way determined by the quadratic residuosity of w. In fact, suppose that $w \in QR_x$. Then obviously $\mathscr{B}(x, y, w, 0) = 1$ (and thus the prover can (x, y)-open w as a 0) and $\mathscr{B}(x, y, w, 1) = 0$, as, by the fact that -1 is a quadratic nonresidue modulo x and by Fact 2.2 $-w \bmod x$ is a quadratic nonresidue modulo x. Now, suppose that $w \in NQR_x$. Then $\mathscr{B}(x, y, w, 1) = 1$, as by the fact that -1 is a quadratic nonresidue modulo x and by Fact 2.13 $-w \bmod x$ is a quadratic residue modulo x, (and thus the prover can (x, y)-open w as a 1) and obviously $\mathscr{B}(x, y, w, 0) = 0$. In our protocol, the (x, y)-opening of w as b is done in a zero-knowledge fashion by using the proof system (C, D) of the previous section. More precisely, as suggested in Remark 3.5, $\mathscr{B}(x, y, w, b)$ is proved to hold by running C on input $(x, (-1)^{1-b} w \bmod x, -y \bmod x)$.

4.1.2. An informal description

Let us now informally describe our proof system. Let $(x, \vec{y}) \in \mathrm{T}(k, m)$ and let $|x| = n$ and $\vec{y} = (y_1, \ldots, y_m)$. First P proves that $x \in \mathrm{BL}$ by running the algorithm A on input x and using a first part of the reference string η. Then, from the reference string η the

prover picks $m\lceil \log(m+1)\rceil$ integers $\rho_{ij} \in Z_x^{+1}$ and a bit b and (x, y_j)-opens each ρ_{ij} as a bit s_{ij} in such a way that the following condition is satisfied: denoted by S_j the integer whose binary representation is $s_{1j} \cdots s_{\lceil \log(m+1)\rceil j}$, the m-tuple $(S_1, \ldots, S_m)$ represents a (k, m)-sequence of admissible shares for b. Now, why is this a proof of the fact that less than k elements $\vec{y}$ are quadratic nonresidues?

Let I be the set of i such that $y_i \in NQR_x$. Then the value of S_i is fixed for all $i \in I$. Thus, if $|I| < k$ then it is always possible to choose S_i for $i \notin I$ such that $(S_1, \ldots, S_m)$ is a sequence of admissible shares for b. Suppose now that $|I| \geqslant k$. Then the values S_i for which $i \in I$ completely determine S. Moreover, the S_i's are uniformly distributed and thus the probability that $S = b$ is at most $1/|\mathscr{E}| \leqslant 1/m$. Thus, the probability that the prover convinces the verifier can be made negligible by repeating the protocol on different parts of the reference string.

A formal description of the proof system (P, V) can be found in Figs. 4 and 5. Here (C, D) is a non-interactive perfect ZK proof system for OR. We denote by h the max of $|x|$ and $|\vec{y}|$. We recall that the length of the random string needed by (C, D) for modulus of length n is $60n^2$. Our field $\mathscr{E}$ is the field with $2^{\lceil \log(m+1)\rceil}$ elements.

Theorem 4.2. (P, V) *is a noninteractive proof system for the language* $T(k, m)$.

Proof. First of all V runs in polynomial time. In fact so do the programs B and D as seen in the previous sections, and step V.3 can be performed in polynomial time [28].

4.1.3. Completeness

If P and V are honest and the number of quadratic nonresidues is less than k, then, thanks to the already discussed properties of the predicate $\mathscr{P}$, at most $k-1$ values S_i are fixed by the random string and cannot be (x, y_j)-opened in a different way by the prover. Moreover, the prover can completely handle the remaining $m-k+1$ values in such a way that the m-tuple $(S_1, \ldots, S_m)$ constitutes a sequence of admissible shares for the bit b.

4.1.4. Soudness

First, note that V halts at step V.2 with negligible probability regardless of whether $(x, \vec{y})$ belongs to $T(k, m)$ or not. In fact, a random n-bit integer belongs to Z_x^{+1} with probability at least $1/8$. Thus the probability that $\rho_{ij} \in Z_x^{+1}$ for less than $mn\lceil \log(m+1)\rceil$ indices is, by Chernoff bound, at most $e^{-mn\lceil \log(m+1)\rceil}$. Therefore the probability that there exists an n-bit modulus x for which this happens is at most $2^n e^{-mn\lceil \log(m+1)\rceil}$, which is negligible.

Suppose $(x, \vec{y}) \notin T(k, m)$; i.e. the number of quadratic nonresidues in $\vec{y}$ is at least k. Then, thanks to the already discussed properties of the predicate $\mathscr{B}$, there exist at least k indices i, $1 \leqslant i \leqslant m$, such that each ρ_{ij}, for $j = 1, \ldots, \lceil \log(m+1)\rceil$, can be $(x, \vec{y}_j)$-opened in just one way by P. Thus, there are at least k values S_i that cannot be opened in a different way by P and the probability that V accepts is the probability that the k values S_i represent a sequence of admissible shares for the bit b, which is at most

Input to P and V:

- A $(h(1+10n^2m\lceil\log(m+1)\rceil+60n^2m\lceil\log(m+1)\rceil))$-bit random string η.
- $(x,\vec{y})\in T(k,m)$, where $|x|=n$ and $\vec{y}=(y_1,\ldots,y_m)$.

Instructions for P.

P.0. (*Divide the reference string and prove that x is a Blum integer.*)

Set $\eta=\tau\circ\sigma^1\circ\cdots\circ\sigma^h$, where $|\tau|=60n^2$ and $|\sigma^l|=1+10n^2m\lceil\log(m+1)\rceil+60n^2m\lceil\log(m+1)\rceil$, for $1\leqslant l\leqslant h$. Run A's algorithm on input x using the random string τ and send its output *Pf*.

Phase $l=1,\ldots,h$:

P.1. (*Prepare the reference string.*)

Let $\sigma^l=b^l\circ\rho^l\circ\gamma^l_{11}\circ\cdots\circ\gamma^l_{1m}\circ\cdots\circ\gamma^l_{\lceil\log(m+1)\rceil 1}\circ\cdots\circ\gamma^l_{\lceil\log(m+1)\rceil m}$, where $|b^l|=1$, $|\rho^l|=10mn^2\lceil\log(m+1)\rceil$ and $|\gamma^l_{ij}|=60n^2$, for $1\leqslant i\leqslant\lceil\log(m+1)\rceil$, $1\leqslant j\leqslant m$.

(For sake of compact notation we drop the superscript l.)

Divide ρ into n-bit integers and denote by ρ_{ij}, $1\leqslant i\leqslant\lceil\log(m+1)\rceil$ and $1\leqslant j\leqslant m$, the first $m\lceil\log(m+1)\rceil$ such integers belonging to Z_x^{+1}. If there are less than $mn\lceil\log(m+1)\rceil$ elements belonging to Z_x^{+1} then HALT.

P.2. (*Construct the sequence of admissible shares.*)

For j such that $y_j\in NQR_x$,

for $i=1,\ldots,\lceil\log(m+1)\rceil$,

if $\rho_{ij}\in QR_x$ then set $s_{ij}\leftarrow 0$,

else set $s_{ij}\leftarrow 1$;

let S_j be the integer whose binary representation is $s_{1j},\ldots,s_{\lceil\log(m+1)\rceil j}$.

For j such that $y_j\in QR_x$,

randomly choose S_j, in such a way that $(S_1,\ldots,S_m)$ constitutes a (k,m)-sequence of admissible shares for the bit b;

let $s_{1j},\ldots,s_{\lceil\log(m+1)\rceil j}$ be the binary representation of S_j.

P.3. (*Prove the correctness of the sequence of admissible shares.*)

For $i=1,\ldots,\lceil\log(m+1)\rceil$,

for $j=1,\ldots,m$,

(x,y_j)-open ρ_{ij} as s_{ij} running the program of C on input $(x,(-1)^{1-s_{ij}}\rho_{ij}\bmod x,\ -y_j\bmod x)$ using γ_{ij} as random string and obtaining as output Π_{ij}.

Send s_{ij} and Π_{ij}.

Fig. 4. The prover P for $T(k,m)$.

$1/|\mathscr{E}|\leqslant 1/m$. Repeating the protocol h times makes the probability of accepting exponentially low. □

4.1.5. Perfect zero-knowledge

For the perfect zero-knowledge property, we have to show an efficient simulator *Sim* such that, for $(x,\vec{y})\in T(k,m)$, the probability space $Sim(x,\vec{y})$ is identical to

Input to V:

- A proof Pf that $x \in BL$.
- A sequence of shares $(S_1^1, \ldots, S_m^1), \ldots, (S_1^h, \ldots, S_m^h)$.
- A sequence of proofs Π_{ij}^l, for $1 \leqslant i \leqslant \lceil \log(m+1) \rceil$, $1 \leqslant j \leqslant m$, $1 \leqslant l \leqslant h$.

Instructions for V.

V.0. (*Divide the reference string and verify that x is a Blum integer.*)
Set $\eta = \tau \circ \sigma^1 \circ \cdots \circ \sigma^h$, where $|\tau| = 60n^2$ and $|\sigma^l| = 1 + n^2 m \lceil \log(m+1) \rceil + 600n^2 m \lceil \log(m+1) \rceil$, for $1 \leqslant l \leqslant h$. Run B's algorithm on input x and τ thus verifying Pf.
Verification of phase $l = 1, \ldots, h$:

V.1. (*Prepare the reference string.*)
Let $\sigma^l = b^l \circ \rho^l \circ \gamma_{11}^l \circ \cdots \circ \gamma_{1m}^l \circ \cdots \circ \gamma_{\lceil \log(m+1) \rceil 1}^l \circ \cdots \circ \gamma_{\lceil \log(m+1) \rceil m}^l$, where $|b^l| = 1$, $|\rho^l| = 10mn^2 \lceil \log(m+1) \rceil$ and $|\gamma_{ij}^l| = 600n^2$, for $1 \leqslant i \leqslant \lceil \log(m+1) \rceil$, $1 \leqslant j \leqslant m$.
(For sake of compact notation we drop the superscript l.)
Divide ρ into n-bit integers and denote by ρ_{ij}, $1 \leqslant i \leqslant \lceil \log(m+1) \rceil$ and $1 \leqslant j \leqslant m$, the first $m \lceil \log(m+1) \rceil$ such integers belonging to Z_x^{+1}. If there are less than $mn \lceil \log(m+1) \rceil$ such integers belonging to Z_x^{+1} then HALT and ACCEPT.

V.2. (*Verify the admissibility of the sequence of shares.*)
Verify that the m-tuple $(S_1, \ldots, S_m)$ is a (k, m)-sequence of admissible shares for the bit b.

V.3. (*Verify that the sequence of admissible shares has been correctly constructed.*)
For $i = 1, \ldots, \lceil \log(m+1) \rceil$,
for $j = 1, \ldots, m$,
verify that the proof Π_{ij} is correct by running the program of D on input $(x, (-1)^{1-s_{ij}} \rho_{ij} \bmod x, -y_j \bmod x)$ using γ_{ij} as random string.
If all verifications are successful then ACCEPT else REJECT.

Fig. 5. The verifier V for $T(k, m)$.

$View_V(x, \vec{y})$. In Fig. 6, we present an efficient algorithm S that simulates one phase of the protocol. (Here F is the simulator of the proof system (C, D) of Section 3). It is straightforward to construct *Sim* using S and M, the simulator of the proof system (A, B) of Section 3.

In the following lemma we prove that $S(x, \vec{y})$ is the same probability space of *Phase* $View_V(x, \vec{y})$, the view of V in one phase of the protocol (P, V).

Lemma 4.3. S *runs in probabilistic polynomial time and for all* $(x, \vec{y}) \in T(k, m)$,

$$S(x, \vec{y}) = PhaseView_V(x, \vec{y}).$$

Proof. Obviously, b has the same distribution in both probability spaces. Also $(S_1, \ldots, S_m)$ is a uniformly distributed sequence of admissible shares of b in both probability spaces.

Instructions for S

Input: $(x, \vec{y}) \in T(k, m)$.

1. Set Proof=empty string.
2. Randomly choose a bit b.
3. Randomly choose $(S_1, \ldots, S_m)$, a (k, m)-sequence of admissible shares for the bit b.
   For $j = 1, \ldots, m$,
   let $s_{1j}, \ldots, s_{\lceil \log(m+1) \rceil j}$ be the binary representation of S_j.
4. For $i = 1, \ldots, 10mn\lceil \log(m+1) \rceil$,
   randomly choose an n-bit integer r_i.
5. If $r_i \in Z_x^{+1}$ for less than $mn\lceil \log(m+1) \rceil$ indices i then **Output:** $(r_1 \circ \cdots \circ r_{10mn\lceil \log(m+1) \rceil})$ and HALT;
6. Set $t \leftarrow 0$.
7. For $i = 1, \ldots, \lceil \log(m+1) \rceil$,
   for $j = 1, \ldots, m$,
   let $\bar{t}$ be the smallest integer $> t$ such that $r_{\bar{t}} \in Z_x^{+1}$;
   toss a fair coin;
   if HEAD set $r_{\bar{t}} \leftarrow y_j^{-s_{ij}} r_{\bar{t}}^2 \bmod x$;
   if TAIL set $r_{\bar{t}} \leftarrow -y_j^{1-s_{ij}} r_{\bar{t}}^2 \bmod x$;
   set $t = \bar{t}$ and $\rho_{ij} \leftarrow r_{\bar{t}}$;
   run F's algorithm on input $(x, (-1)^{1-s_{ij}} \rho_{ij} \bmod x, -y_j \bmod x)$ obtaining as output (γ_{ij}, Π_{ij});
   append Π_{ij} to Proof.
8. Set $\rho = r_1 \circ \cdots \circ r_{10mn\lceil \log(m+1) \rceil}$.

Output: $(b \circ \rho \circ \gamma_{11} \circ \cdots \circ \gamma_{1m} \circ \cdots \circ \gamma_{\lceil \log(m+1) \rceil 1} \circ \cdots \circ \gamma_{\lceil \log(m+1) \rceil m}$, Proof).

Fig. 6. The simulator S.

Let us now take a look at the strings ρ_{ij}'s. We have two different cases, depending on the quadratic residuosity of y_j.

(1) $y_j \in NQR_x$: In the view of the verifier the values of s_{ij} are determined by the quadratic residuosity modulo x of ρ_{ij}. More precisely, $s_{ij} = 1$ if and only if ρ_{ij} is a quadratic nonresidue. In the output of S, if $s_{ij} = 1$, the ρ_{ij} is either equal to $r_{\bar{t}}^2 y_j^{-1} \bmod x$ (which by Fact 2.2 is a quadratic nonresidue) or to $-r_{\bar{t}}^2 \bmod x$ (which by Fact 2.7 is a quadratic nonresidue). On the other hand, if $s_{ij} = 0$, ρ_{ij} is either equal to $r_{\bar{t}}^2 \bmod x$ (which is a quadratic residue) or to $-y_j r_{\bar{t}}^2 \bmod x$ (which by Facts 2.2 and 2.7 is a quadratic residue).

(2) $y_j \in QR_x$. As both $\mathcal{B}(x, y, \rho_{ij}, 0) = 1$ and $\mathcal{B}(x, y, \rho_{ij}, 1) = 1$, P can choose each s_{ij}, i.e. the (x, y_j)-opening of each ρ_{ij}, either as 0 or as 1 in order to correctly construct a uniformly distributed sequence of admissible shares. Thus, in the view of the verifier, the value of s_{ij} does not depend on the quadratic residuosity of ρ_{ij}, and ρ_{ij} is a quadratic residue or a quadratic nonresidue with probability 1/2. In the output of S,

the s_{ij}'s are chosen in order to form a uniformly distributed sequence of admissible shares. Moreover, if $s_{ij}=1$, then with probability 1/2 the element ρ_{ij} is set equal to $r_i^2 y_j^{-1} \bmod x$ (which by Fact 2.2 is a quadratic residue) and with probability 1/2 the element ρ_{ij} is set equal to $-r_i^2 \bmod x$ (which by Fact 2.7 is a quadratic nonresidue). On the other hand, if $s_{ij}=0$, with probability 1/2 the element ρ_{ij} is set equal to $-r_i^2 y_j \bmod x$ (which by Facts 2.2 and 2.7 is a quadratic nonresidue) and with probability 1/2 the element ρ_{ij} is set equal to $r_i^2 \bmod x$ (which is a quadratic residue).

Moreover, by construction, ρ is uniformly distributed over all binary strings of the appropriate length. Finally, the strings γ_{ij} and the "proofs" Π_{ij} have the same distribution since (C, D) is a perfect ZK proof system. □

The above lemma and Theorem 4.2 prove the following theorem.

Theorem 4.4. (P, V) *is a noninteractive perfect zero-knowledge proof system for the language* $\mathrm{T}(k, m)$.

Remark 4.5. The protocol (P, V) that we have just presented is designed for the case when the quadratic residuosities of the y_i's are considered modulo the same integer x. However, it can be easily seen that this restriction can be removed. That is, for all $1 \leqslant k \leqslant m$, the language

$$\mathrm{MT}(k,m)=\{(\vec{x},\vec{y}) \mid x_i \in \mathrm{BL},\ y_i \in Z_x^{+1},\ \text{for } i=1,\ldots,m,$$

$$\text{and } |\{y_i \mid y_i \in NQR_{x_i}\}| < k\}$$

also has a noninteractive perfect zero-knowledge proof system. In this proof system, the prover runs P's algorithm with the following modifications. First of all at step P.0 he runs algorithm A m times using different pieces of the reference string, to prove that each x_i is a Blum integer. Then, whenever the modulus x is used, P uses the modulus x_j, if operations on y_j have to be performed. Note, that it is never the case that in a step, P performs operations relative to two different y_i's. These modifications are also reflected in similar modifications in the algorithms for the verifier and the simulator.

Remark 4.6. The protocol (P, V) can be also extended to a proof system for the language

$$\overline{\mathrm{MT}}(k,m)=\{(\vec{x},\vec{y}) \mid x_i \in \mathrm{BL},\ y_i \in Z_x^{+1},\ \text{for } i=1,\ldots,m,$$

$$\text{and } |\{y_i \mid y_i \in NQR_{x_i}\}| \geqslant k\},$$

where $1 \leqslant k \leqslant m$. The new prover uses the algorithm P with the modifications of the previous remark on input the pair $(\vec{x},\vec{y})$ where $z_i = -y_i \bmod x_i$, for $i=1,\ldots,m$.

5. Round-optimal interactive perfect zero knowledge proof systems

The aim of this section is to present a general procedure that, for languages with some specific properties, transforms a noninteractive ZK proof system into a 4-round ZK proof system. The transformation preserves perfect zero knowledgeness; i.e. if the noninteractive proof system we started with is perfect zero knowledge, then we obtain a perfect zero-knowledge interactive proof system (for complete definitions of zero-knowledge interactive proof systems, we refer the reader to [25]).

In the next subsection, for sake of exposition, we describe and analyze the transformation for the language $T(k,m)$ and, then, briefly discuss the extension to any language with the required properties.

5.1. *Round-optimal perfect zero knowledge for* $T(k,m)$

In the previous section we have shown a noninteractive perfect ZK proof system (P, V) for $T(k,m)$. We use this proof system in order to create an interactive 4-round perfect zero-knowledge proof system (PROVER, VERIFIER). Thus, unless the language $T(k,m)$ is in BPP, (PROVER, VERIFIER) is round optimal, thanks to a result of [19].

Let us give an informal description of (PROVER, VERIFIER). The first three rounds are used by PROVER and VERIFIER to randomly select a string σ in the following way. First, VERIFIER commits to his random bits; then PROVER sends his random bits, and finally VERIFIER open his commitments. The reference string σ will be formed by the xoring of PROVER's random bits, with the bits decommitted by VERIFIER. Then, in the fourth round PROVER runs P's program on input $(x,\vec{y})$ using σ as a reference string. Finally VERIFIER verifies that the string σ has been correctly computed and runs V's program on input $(x,\vec{y})$ and σ. The following property of Blum integers will be useful for constructing a bit commitment.

Lemma 5.1. *Let* $x \in \mathrm{BL}$, $w \in QR_x$ *and let* $\pm r_1 \bmod x$ *and* $\pm r_2 \bmod x$ *be its* 4 *square roots. Then* $(r_1 | x) = -(r_2 | x)$.

Proof. Suppose that $r_1, r_2 \leqslant x/2$ and write x as $x = p^{k_1} q^{k_2}$, where k_1 and k_2 are odd (by definition of Blum integers); from $r_1^2 \equiv r_2^2 \bmod x$ it follows that $(r_1 - r_2)(r_1 + r_2) = hx$ for some $h < x$. In general we can write $r_1 - r_2 = h_1 p^{i_1} q^{j_1}$, and $r_1 + r_2 = h_2 p^{i_2} q^{j_2}$, where $h_1 h_2 = h$, $i_1 + i_2 = k_1$, $j_1 + j_2 = k_2$, and $i_1, i_2, j_1, j_2 \geqslant 0$. We see that $i_1 = j_2 = 0$ or $j_1 = i_2 = 0$. Suppose that it is not so, then $2r_1 = h_1 p^{i_1} q^{j_1} + h_2 p^{i_2} q^{j_2}$, and so $p | r_1$ or $q | r_1$, that is a contradiction as $r_1 \in Z_x^*$. Assume $j_1 = i_2 = 0$ (the other case being similar); then we can write $r_1 - r_2 = h_1 p^{k_1}$ and $r_1 + r_2 = h_2 q^{k_2}$. Moreover, from $p | (r_1 - r_2)$ and $q | (r_1 + r_2)$, we have $r_1 \equiv r_2 \bmod p$ and $r_1 \equiv -r_2 \bmod q$. From this it follows that $(r_1 | p) = (r_2 | p)$ and $(r_1 | p) = -(r_2 | q)$ and, finally, $(r_1 | x) = -(r_2 | x)$. □

Using Lemma 5.1, the commitment can be implemented as follows. To commit to a bit b, it is sufficient for VERIFIER to choose randomly an $r \in Z_x^*$ such that r has Jacobi symbol -1 (if $b = 0$) or $+1$ (if $b = 1$) and to give $w = r^2 \bmod x$ to PROVER.

A decommitment is done by simply revealing r. Given only w, PROVER cannot compute b better than guessing at random. In fact, by Lemma 5.1 there exist two square roots of w that have different Jacobi symbols and PROVER does not know which one will be revealed later by VERIFIER. On the other side, if VERIFIER is able to reveal the decommitment in two ways, then he is able to factor x (see Lemma 5.2 below).

Let $l(m, n)$ be the length of the reference string used in the proof system (P, V) on input $(x, \vec{y})$, with $|x| = n$ and $|\vec{y}| = m$. A formal description of (PROVER, VERIFIER) can be found in Fig. 7.

The completeness and soundness of the above protocol follow directly from the completeness and soundness of (P, V).

To prove the perfect zero-knowledge property of the proof system (PROVER, VERIFIER), we show an efficient simulator SIMUL such that, on $(x, \vec{y}) \in \mathrm{T}(k, m)$ and interacting with a possibly malicious verifier VERIFIER', such that the probability space SIMUL$(x, \vec{y})$ is equal to the view of VERIFIER' in the protocol. The simulator SIMUL is based on the *double running* technique [3]. First, the simulator *Sim* of the proof system (P, V) is run and then two strings σ and PROOF are obtained. Then SIMUL performs the protocol until step VERIFIER'.2 where he learns the bit committed to by VERIFIER'. At this point, he rewinds VERIFIER' in the state just after step VERIFIER'.1 and chooses the bits b_i in such a way that he obtains the string σ at step PROVER.2. If, during the second execution, VERIFIER' opens one of his commitments in a different way (i.e. he gives a different square root of some w_i's), then SIMUL can factor x (see Lemma 5.2 below) and thus he can run P's program (that can be executed in polynomial time

---

The Proof System (PROVER, VERIFIER)

Input to PROVER **and** VERIFIER: $(x, \vec{y}) \in \mathrm{T}(k, m)$, where $|x| = n$, and $\vec{y} = (y_1, \ldots, y_m)$.

VERIFIER.1: For $i = 1, \ldots, l(m, n)$,
randomly choose $r_i \in Z_x^*$, set $w_i = r_i^2 \bmod x$, and send w_i.

PROVER.1: For $i = 1, \ldots, l(m, n)$,
randomly choose and send $b_i \in \{0, 1\}$.

VERIFIER.2: For $i = 1, \ldots, l(m, n)$,
compute $c_i = ((r_i | x) + 1)/2$ and send r_i, c_i.

PROVER.2: For $i = 1, \ldots, l(m, n)$,
if $r_i^2 = w_i \bmod x$ and $(r_i | x) = 2c_i - 1$ then continue, else HALT;
set $\sigma_i = b_i \oplus c_i$;
set $\sigma = \sigma_1 \circ \cdots \circ \sigma_{l(m,n)}$ and run P's program on input $(x, \vec{y})$, using σ as reference string. Send its output PROOF.

VERIFIER.3: Verify that σ is correctly computed and run V's program on input $x, \vec{y}, \sigma$, PROOF.
If all verifications are successful then ACCEPT else REJECT.

Fig. 7. The proof system (PROVER, VERIFIER).

since x's factorization is available) and perfectly simulate the interaction between P and V.

Let $h=\max(|x|,|\vec{y}|)$ and let c be a constant such that the running time of VERIFIER is bounded above by the polynomial h^c. A formal description of program SIMUL is in Fig. 8.

Proof. First note that the Jacobi symbol can be computed in probabilistic polynomial time, by Fact 2.5. Then it is easy to see that steps SIMUL.1–SIMUL.11 are trivially performed in probabilistic polynomial time. In step SIMUL.12 x's factorization has to be computed. However, when entering in this step, SIMUL has obtained from VERIFIER′ two numbers $r_1, r_2 \in Z_x^*$ such that $(r_1|x)=-(r_2|x)$, and $r_1^2 \equiv r_2^2 \bmod x$. Thus, it only remains to show that this information is sufficient for SIMUL to compute x's factorization in polynomial time. First of all note that, as x is a Blum integer, for all $z \in Z_x^*$ we have that $(z|x)\equiv(-z|x)$. Thus, as r_1 and r_2 have different Jacobi symbols, it must be the case that $r_1 \not\equiv -r_2 \bmod x$. By the proof of Lemma 5.1 one gets $r_1 - r_2 = h_1 p^{k_1}$, and $r_1 + r_2 = h_2 q^{k_2}$. Then $\gcd(r_1-r_2, x)=\gcd(h_1 p^{k_1}, p^{k_1} q^{k_2})=p^{k_1}\gcd(h_1, q^{k_2})=p^{k_1}q^d$ for some $d \geqslant 0$, and $\gcd(r_1+r_2, x)=\gcd(h_2 q^{k_2}, p^{k_1} q^{k_2})=q^{k_2}\gcd(h_2, p^{k_1})=p^a q^{k_2}$ for some $a \geqslant 0$. Note that if $d>0$ or $a>0$, then $q|h_1$ or $p|h_2$, and from the relation $2r_1 = h_1 p^{k_1} + h_2 q^{k_2}$ we have that $q|r_1$ or $p|r_1$, which is a contradiction, as $r_1 \in Z_x^*$. Thus, $\gcd(r_1+r_2, x)$ and $\gcd(r_1-r_2, x)$ are prime powers, and so it is easy now from these two numbers to efficiently compute the primes p and q that factor x, as in Section 3. □

Lemma 5.3. *For each* $(x, \vec{y}) \in \mathrm{T}(k, m)$, *the probability space* SIMUL $(x, \vec{y})$ *is equal to the view of* VERIFIER′ *in the protocol* (PROVER, VERIFIER) *on input* $(x, \vec{y})$.

Proof. First it is easy to see that both in the simulator and in the protocol: (i) $w_1, \ldots, w_{l(m,n)}$ are random quadratic residues modulo x; (ii) $w_i = r_i^2 \bmod x$ for $i=1, \ldots, l(m,n)$. Then we see that the bits $b_1, \ldots, b_n$ are uniformly distributed. In fact in step SIMUL.8 each b_i is set equal to $\sigma_i \oplus c_i$. From the perfect zero knowledgeness of the simulator *Sim* for the noninteractive ZK proof system (P, V) it follows that each σ_i is uniformly distributed over $\{0,1\}$ and so is also b_i. The string PROOF can be generated by the simulator SIMUL in two ways: (a) by the simulator *Sim* for (P, V); in this case, from the perfect zero knowledgeness of *Sim*, it follows that PROOF has the same distribution both in SIMUL and in (PROVER, VERIFIER); or (b) by running P's program, once obtained x's factorization; in this case, it is obvious that PROOF is generated by SIMUL exactly in the same way as in (PROVER, VERIFIER). Also the reference string σ can be generated by the simulator SIMUL in two ways: (a′) by the simulator *Sim* for (P, V); in this case, from the perfect zero knowledgeness of *Sim*, it follows that σ is uniformly distributed over $\{0,1\}^{l(m,n)}$, as in (PROVER, VERIFIER); (b′) at step SIMUL.12 by xoring each uniformly distributed bit b_i' with the bit c_i' given by VERIFIER′ and then concatenating the bits σ_i thus obtained. Finally, each σ_i is correctly computed as the xoring of a bit given by VERIFIER′ and a random bit given by SIMUL

Instructions for SIMUL.

Input: $(x, \vec{y}) \in T(k, m)$.

SIMUL.1. Randomly choose a h^c-bit string R and bits $b_1, \ldots, b_{l(m,n)}$;
write R on the random tape of VERIFIER'.

SIMUL.2. Run *Sim* on input $(x, \vec{y})$ obtaining as output (σ, PROOF);
let $\sigma = \sigma_1 \circ \cdots \circ \sigma_{l(m,n)}$.

SIMUL.3. Set change-of-dec $\leftarrow$ no.

SIMUL.4. For $i = 1, \ldots, l(m, n)$,
get w_i from VERIFIER'.

SIMUL.5. For $i = 1, \ldots, l(m, n)$,
write b_i on the communication tape of VERIFIER'.

SIMUL.6. For $i = 1, \ldots, l(m, n)$,
get r_i from VERIFIER'.

SIMUL.7. If $r_i^2 \neq w_i \bmod x$ for some $i \in \{1, \ldots, l(m, n)\}$ then
output $(R, \vec{w}, \vec{b}, \vec{r})$ and HALT.

SIMUL.8. For $i = 1, \ldots, l(m, n)$,
set $c_i = ((r_i | x) + 1)/2$ and $b_i' = \sigma_i \oplus c_i$ (so that $\sigma_i = b_i' \oplus c_i$).

SIMUL.9. Rewind VERIFIER' to the state just after step SIMUL.4;
for $i = 1, \ldots, l(m, n)$,
write b_i' on the communication tape of VERIFIER'.

SIMUL.10. For $i = 1, \ldots, l(m, n)$,
get r_i' from VERIFIER';
if $r_i'^2 \neq w_i \bmod x$ for some $i \in \{1, \ldots, l(m, n)\}$ then
output $(R, \vec{w}, \vec{b}', \vec{r}')$ and HALT;
if $r_i \neq r_i'$ then
set change-of-dec $\leftarrow$ yes.

SIMUL.11. If change-of-dec = no then output $(R, \vec{w}, \vec{b}', \vec{r}', \sigma, \text{PROOF})$ and HALT.

SIMUL.12. If change-of-dec = yes then
compute x's factorization;
for $i = 1, \ldots, l(m, n)$,
set $c_i' = ((r_i | x) + 1)/|2$, and $\sigma_i' = b_i' \oplus c_i'$;
let $\sigma' = \sigma_1' \circ \cdots \circ \sigma_{l(m,n)}'$;
run P's program on input $(x, \vec{y})$ and σ' using x's factorization and obtaining PROOF;
output $(R, \vec{w}, \vec{b}', \vec{r}', \sigma', \text{PROOF})$ and HALT.

Fig. 8. The simulator SIMUL.

while simulating PROVER. In fact, in case (a') at step SIMUL.8 the bit b_i' is set equal to $\sigma_i \oplus c_i$ and in case (b') at step SIMUL.12 the bit σ_i is set equal to $b_i' \oplus c_i'$. □

The above lemmas prove the following theorem.

Theorem 5.4. (PROVER, VERIFIER) *is a 4-round perfect zero knowledge proof system for the language* $T(k,m)$.

5.2. Extensions

Looking at the properties of $T(k,m)$ that we used in the above protocol, we can generalize it for other languages. In fact we can define a class of quadratic residuosity languages L that satisfy the following properties:
there exists a predicate $R(\vec{x},\vec{y})$ such that

(1) $L=\{(\vec{x},\vec{y})\,|\,x_i\in BL$, for $i=1,\ldots,m$, and $R(\vec{x},\vec{y})=1\}$.

(2) There exists a noninteractive ZK proof system (P,V) for L in which the prover runs in probabilistic polynomial time if given the factorization of all the x_i's.

First of all, note that the protocol (PROVER, VERIFIER) would give a round-optimal perfect zero-knowledge proof system for the language L if the Blum integers x_i were all equal. On the other hand, some modifications are necessary to handle the most general case in which the x_i can be different. More precisely, the commitment made by the verifier is modified in the following way. To commit to a bit b, the verifier randomly chooses an $r_i\in Z^*_{x_i}$, for $i=1,\ldots,m$, such that each Jacobi symbol $(r_i\,|\,x_i)$ is equal to -1 (if $b=0$) or to $+1$ (if $b=1$) and gives $w_i=r_i^2 \bmod x_i$, for $i=1,\ldots,m$, to the prover. A decommitment is done by simply revealing each r_i. Similarly to the commitment scheme of (PROVER, VERIFIER), given only the w_i's, the prover cannot compute b better than guessing at random. On the other side, if the verifier is able to reveal the decommitment in two ways, then he reveals two square roots with the different Jacobi symbols of each w_i modulo x_i. As before, with this information it is possible to factor each x_i and thus the simulator can run P's algorithm on a truly random string. Given such a commitment scheme, the rest of the protocol is easily obtained as in the case of $T(k,m)$.

Like (PROVER, VERIFIER), also this protocol preserves perfect zero knowledgeness; i.e. if (P, V) is perfect zero knowledge, so is also the resulting 4-round protocol described. In particular we obtain a round-optimal interactive perfect ZK proof system for all the languages seen as far, for they all belong to this class.

5.3. An open problem

The technique based on threshold schemes for proving a threshold gate does not seem to generalize to the case of arbitrary threshold circuits. Thus the problem of obtaining a noninteractive perfect zero-knowledge proof system for the language of threshold circuits is open.

Acknowledgment

We thank an anonymous referee for careful reading and useful remarks on the paper.

References

[1] D. Angluin, Lecture notes on the complexity of some problems in number theory, Tech. Report 243, Yale Univ. 1982.

[2] D. Angluin and L. Valiant, Fast probabilistic algorithms for Hamiltonian circuits and matchings, *J. Comput. System Sci.* **18** (2) (1979) 155–193.
[3] M. Bellare, S. Micali and R. Ostrovsky, Perfect zero knowledge in constant rounds, in: *Proc. 22th Ann. Symp. on the Theory of Comput.* (1990) 482–493.
[4] M. Bellare and M. Yung, Certifying cryptographic tools: The case of trapdoor permutations, in: *Proc. of CRYPTO'92*, to appear.
[5] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Hastad, S. Micali and P. Rogaway, Everything provable is provable in zero knowledge, in S. Goldwasser, ed., *Advances in Cryptology – CRYPTO 88*, Lecture Notes in Computer Science, Vol. 403 (Springer, Berlin) 37–56.
[6] G.R. Blackley, Safeguarding cryptographic keys, in: *Proc. AFIPS 1979 National Computer Conf.*, (1979) 313–317.
[7] M. Blum, A. De Santis, S. Micali and G. Persiano, Non-interactive zero-knowledge, *SIAM J. Comput.* **20** (1991) 1084–1118.
[8] M. Blum, P. Feldman and S. Micali, Non-interactive zero-knowledge and applications, in: *Proc. 20th Ann. ACM Symp. on Theory of Comput.* (1988) 103–112.
[9] R. Boppana, Amplification of probabilistic boolean formulas, *Adv. Comput. Res.* **5** (1989) 27–45.
[10] R. Boppana, J. Hastad and S. Zachos, Does co-NP has short interactive proofs?, *Inform. Process Lett.* **25** (1987) 127–132.
[11] J. Boyar, K. Friedl and C. Lund, Practical zero-knowledge proofs: giving hints and using deficiencies, *J. Cryptology*, **4** (1991) 185–206.
[12] G. Brassard, D. Chaum and C. Crépeau, Minimum disclosure proofs of knowledge, *J. Comput. System Sci.* **37** (2) (1988) 156–189. (Preliminary version in *FOCS '86.*)
[13] G. Brassard, C. Crépeau and M. Yung, Perfect zero-knowledge computationally convincing proofs for NP in constant rounds, *Theoret. Comput. Sci.* **84** (1991) 23–52.
[14] A. De Santis, S. Micali and G. Persiano, Non-interactive zero-knowledge proof-systems, in: *Advances in Cryptology – CRYPTO 87*, Lecture Notes in Computer Science, Vol. 293 (Springer, Berlin) 52–72.
[15] A. De Santis, G. Persiano and M. Yung, Perfect zero-knowledge proofs for graph isomorphism languages, manuscript.
[16] P. Erdős and J. Spencer, *Probabilistic Methods in Combinatorics* (Academic Press, New York, 1974).
[17] U. Feige, D. Lapidot and A. Shamir, Multiple noninteractive zero-knowledge proofs based on a single random string, in: *Proc. 22nd Ann. Symp. on the Theory of Comput* (1990) 308–317.
[18] L. Fortnow, The complexity of perfect zero knowledge, in: *Proc. 19th Ann. ACM Symp. on Theory of Comput.* (1987) 204–209.
[19] O. Goldreich and H. Krawczyk, On the composition of zero-knowledge proof systems, in: *Proc. 17th Internat. Coll. on Automata, Languages and Programming* (1986) 174–187.
[20] O. Goldreich and E. Kushilevitz, A perfect zero knowledge proof for a decision problem equivalent to discrete logarithm, in: S. Goldwasser, ed., *Advances in Cryptology – CRYPTO 88*, Lecture Notes in Computer Science, Vol. 403 (Springer, Berlin) 57–70.
[21] O. Goldreich, S. Micali and A. Wigderson, Proofs that yield nothing but their validity, or all languages in NP have zero-knowledge proofs, *J. ACM* **38** (1991) 691–729.
[22] O. Goldreich, S. Micali and A. Wigderson, How to play any mental game, in: *Proc. 19th An. ACM Symp. on Theory of Comput.* 218–229.
[23] O. Goldreich and E. Petrank, Quantifying knowledge complexity, in: *Proc. 32th IEEE Symp. on Foundations of Comput. Sci.* (1991).
[24] S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput. System. Sci.* **28** (1984) 270–299.
[25] S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive proof-systems, *SIAM J. Comput.* **18** (1989).
[26] R. Impagliazzo and M. Yung, Direct minimum knowledge computations in: *Advances in Cryptology – CRYPTO 87*, Lecture Notes in Computer Science, Vol. 273 (Springer, Berlin) 40–51.
[27] I. Niven and H.S. Zuckerman, *An Introduction to the Theory of Numbers* (Wiley, New York, 1960).
[28] A. Shamir, How to share a secret, *Comm. ACM* **22** (1979) 612–613.
[29] M. Tompa and H. Woll, Random self-reducibility and zero-knowledge interactive proofs of possession of information, in: *Proc. 28th Symp. in Foundations of Comput. Sci.* (1987) 472–482.