Secret Sharing and Perfect Zero Knowledge*

A. De Santis,[1] G. Di Crescenzo,[1] G. Persiano[2]

[1] Dipartimento di Informatica ed Applicazioni,
Università di Salerno, 84081 Baronissi (SA), Italy

[2] Dipartimento di Matematica,
Università di Catania, 95125 Catania, Italy

Abstract. In this work we study relations between secret sharing and perfect zero knowledge in the non-interactive model. Both secret sharing schemes and non-interactive zero knowledge are important cryptographic primitives with several applications in the management of cryptographic keys, in multi-party secure protocols, and many other areas. Secret sharing schemes are very well-studied objects while non-interactive *perfect* zero-knowledge proofs seem to be very elusive. In fact, since the introduction of the non-interactive model for zero knowledge, the only perfect zero-knowledge proof known was for quadratic non residues.
In this work, we show that a large class of languages related to quadratic residuosity admits *non-interactive perfect zero-knowledge proofs.* More precisely, we give a protocol for proving non-interactively and in perfect zero knowledge the veridicity of any "threshold" statement where atoms are statements about the quadratic character of input elements. We show that our technique is very general and extend this result to any secret sharing scheme (of which threshold schemes are just an example).

1 Introduction

Secret Sharing. The fascinating concept of *Secret Sharing* scheme has been first considered in [18] and [3]. A secret sharing scheme is a method of dividing a secret s among a set of participants in such a way that only qualified subsets of participants can reconstruct s but non-qualified subsets have absolutely no information on s. Secret sharing schemes are very useful in the management of cryptographic keys and in multi-party secure protocols (see for instance [13]). For an unified description of recent results in the area of secret sharing schemes, and for a complete bibliography, see [20] and [19].

Zero Knowledge. The seminal concept of a *Zero-Knowledge* proof has been introduced in [15] that gave zero-knowledge proofs for the number-theoretic languages of quadratic residues and quadratic non residues. A Zero-Knowledge (ZK) proof is a special kind of proof that allows an all-powerful prover to convince

* Work supported by MURST and CNR
D.R. Stinson (Ed.): Advances in Cryptology - CRYPTO '93, LNCS 773, pp. 73-84, 1994.

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 [13] that proved that indeed all NP languages 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 (see [14]).

The zero-knowledge proof for all NP of [13], are *computational* zero-knowledge, that is secure only against poly-bounded adversaries, whereas those in [15] are perfect, that is secure against unlimited-power adversaries. Moreover, the proofs of [13] 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, see [7] and [11]) 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 [15, 13, 12, 21, 4].

The shared-string model for Non-Interactive ZK has been put forward in [6] and further elaborated by [8, 5]. In this model, prover and verifier share a random string and the communication is monodirectional. In [5] it is proved that under the quadratic residuosity assumption all NP languages have non-interactive computational zero-knowledge proofs in this model. Subsequently, in [10] it was proved that certified trapdoor permutations are sufficient for proving non-interactively and in zero knowledge membership to any language in NP ([1] relaxed the assumption by proving that trapdoor permutations are sufficient). In the non-interactive model the only perfect zero-knowledge proof has been given in [5] for the language of quadratic non residuosity modulo *Regular*(2) integers.

Because of their importance, obtaining perfect-ZK proofs for certain classes of languages still remains an important research area.

Organization of the paper and Our Results In the next section we review some number-theoretic results about quadratic residuosity and the definition of perfect zero-knowledge in the non-interactive model of [5].

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 non residue modulo x and x is a Blum integer.

In Section 4, we present our main result: a perfect non-interactive zero-knowledge proof system for any threshold statement of quadratic residuosity of any number of inputs. That is, for all $k \leq m$, we give a proof system for the language $\mathrm{T}(k, m)$ of $(m+1)$-uples $(x, y_1, \cdots, y_m)$ such that less than k of the y_i's are quadratic non residues modulo x and x is a Blum integer. Our construction is based upon the properties of secret sharing schemes. We give a way of con-

structing a set of shares from the random string that has the following property. If less than k of the y_i's are quadratic non residues modulo x, then this set can be opened by the prover as a sharing 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 non residues 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 $\mathrm{T}(k, m)$, 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 briefly discuss the generalization of the tecnique of the previous section to statements based on secret sharing schemes in which the subsets of participants that recover the secret are arbitrary. That is, given a secret sharing scheme for an access structure on a set of participants, then we can construct a non-interactive perfect zero-knowledge proof system for a special formula based on the access structure given.

In all of 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.

2 Background and Notations

2.1 Notations

We identify a binary string σ with the integer x whose binary representation 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 $k-$uple $(w_1, \ldots, w_k)$ of numbers or bits. We say that $\vec{w} \in S$ 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. If $\vec{x}$ is a k-uple, by the expression $|\vec{x}|$ we denote the number k of components of $|\vec{x}|$.

2.2 Number Theory

We refer the reader to [17] and [5] for the definitions of quadratic residues, Jacobi symbol and Regular(s) integers. We define the *quadratic residuosity predicate* as $Q_x(y) = 0$ if y is a quadratic residue modulo x and 1 otherwise. Moreover, we let Z_x^{+1} and Z_x^{-1} denote, respectively, the sets of elements of Z_x^* with Jacobi symbol $+1$ and -1 and $QR_x = \{y \in Z_x^{+1} | Q_x(y) = 0\}$, $NQR_x = \{y \in Z_x^{+1} | Q_x(y) = 1\}$.

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

Definition 1. An integer x is a Blum integer, in symbols $x \in \mathrm{BL}$, 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.

From Euler's criterion it follows that, if x is a Blum integer, $-1 \bmod x$ is a quadratic non residue with Jacobi symbol $+1$. Moreover we have the following fact.

Fact 1. On input a Blum integer x, it is easy to generate a random quadratic non residue 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. 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.

On the other hand, if x is a product of two prime powers, but not a Blum integer, then the above lemma does not hold.

Lemma 3. Let $x = p^{k_1}q^{k_2}$, where $p \equiv 1 \bmod 4$. Then, at least one half of the quadratic residues has no square root which is itself a quadratic residue modulo x.

The following characterization of Blum integers will be used to obtain a non-interactive perfect zero-knowledge proof system for the set of Blum integers.

Fact 2. An integer x is a Blum integer if and only if $x \in \mathit{Regular}(2)$, $-1 \bmod x \in NQR_x$, and for each $w \in QR_x$ there exists an r such that $r^4 \equiv w \bmod x$.

2.3 Non-Interactive Perfect Zero Knowledge

Let us now review the definition of Non-Interactive Perfect ZK of [5] (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 P a probabilistic Turing machine and V a deterministic Turing machine that runs in time polynomial in the length of its first input.

Definition 4. We say that (P, V) is a Non-Interactive Perfect Zero-Knowledge Proof System (Non-Interactive 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 $\forall x \in L$, the two probability spaces $S(x)$ and $View_V(x)$ are equal, where by $View_V(x)$ we denote the probability space

$$View_V(x) = \{\sigma \leftarrow \{0,1\}^{|x|^c};\ Proof \leftarrow P(\sigma, x) : (\sigma, Proof)\}.$$

We notice 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 Non-Interactive 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.)

2.4 Secret Sharing

Shamir [18] and Blackley [3] were the first to consider the problem of secret sharing and gave secret sharing schemes known as *threshold schemes.* We review the notion of threshold scheme as it will be instrumental for our construction of a non-interactive perfect zero-knowledge proof system for threshold statements of quadratic residuosity. 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 [18] shows how to construct such threshold schemes using interpolation of polynomials. We have the following fact:

Fact 3. The following is a (k, m)-threshold scheme. Let $(\mathcal{E}, +, \cdot)$ be a finite field with more than m elements and let S be the value to be shared. Choose at random $a_1, \ldots, a_{k-1} \in \mathcal{E}$, 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 $\mathcal{E}$).

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 $\mathcal{E}$, such that $a_0 = S$ and $S_i = q(i)$ for $i = 1, \ldots, m$.

Remark. 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 $i \notin I$ 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 some S, then S is uniformly distributed in $\mathcal{E}$. On the other hand, if $|I| = k$, 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.

3 Preliminary Results

In this section we discuss two simple non-interactive perfect zero-knowledge proof systems for the language BL of Blum integers and for the language OR of logical or of quadratic residuosity that we will define later. They will be useful in the construction of our main result.

3.1 The Proof System for BL

A proof system (A,B) for BL is easily obtained using the characterization of Blum integers given by Fact 2. In fact, it is sufficient for the prover to first prove that x is a $Regular(2)$ integer and that -1 is a quadratic non residue modulo x using the proof system given in [5]. 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.

3.2 The Proof System for OR

We now describe a non-interactive perfect ZK proof system (C,D) for the language

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

This is an extension of the proof system for quadratic non residuosity given in [5].
In our construction we will use the following

Definition 5. For any positive integer x, define the relation $\approx_x$ on $Z_x^{+1} \times Z_x^{+1}$ as follows: $(a_1, a_2) \approx_x (b_1, b_2) \iff a_1 b_1 \in QR_x$ and $a_2 b_2 \in QR_x$.

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). 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.

Let us informally describe the protocol (C,D). By (A,B) we denote the non-interactive perfect zero-knowledge proof system for the language BL described above. On input (x, y_1, y_2), C and D share a reference string $\gamma = \rho \circ \sigma$, where σ is split into pairs $(\sigma_{i,1}, \sigma_{i,2})$. 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 first that x is a Blum integer, by running the algorithm B, and then 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 two situations may happen: (a) $x \notin \mathrm{BL}$ or (b) $x \in \mathrm{BL}$ and $y_1, y_2 \in QR_x$. In the first D accepts with negligible probability because of soundness of the proof system (A,B). In the second 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 non residue, 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 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.

Let us now give a sketch of the proof that (C,D) is perfect zero-knowledge. On input $(x, y_1, y_2) \in \mathrm{OR}$, the simulator S has to generate uniformly distributed pairs $(\sigma_{i,1}, \sigma_{i,2})$ belonging to each of the four classes of $Z_x^{+1} \times Z_x^{+1}$ determined by the relation $\approx_x$. Notice that, on input $(x, y_1, y_2) \in \mathrm{OR}$, it is possible to efficiently construct four pairs $(\alpha_1, \beta_1), \ldots, (\alpha_4, \beta_4)$, each belonging to a different $\approx_x$ class, in the following way. Randomly choose $r, s \in Z_x^*$, and output

$$
\begin{aligned}
(\alpha_1, \beta_1) &= (y_1, y_2), \\
(\alpha_2, \beta_2) &= (r^2 \bmod x, s^2 \bmod x), \\
(\alpha_3, \beta_3) &= (y_1 y_2 \bmod x, y_1), \\
(\alpha_4, \beta_4) &= (y_2, y_1 y_2 \bmod x).
\end{aligned}
$$

Thus it is possible to efficiently generate a uniformly distributed pair belonging to one (randomly chosen) of the four $\approx_x$ classes, in the following way. Randomly choose $j \in \{1, \ldots, 4\}$ and $u, v \in Z_x^*$, and output the pair $(\sigma_{j,1}, \sigma_{j,2})$, where $\sigma_{j,1} = \alpha_i^{-1} u^2 \bmod x$ and $\sigma_{j,2} = \beta_i^{-1} v^2 \bmod x$. Moreover, it is easy for S to give random square roots modulo x of the products $\sigma_{j,1}\alpha_i \bmod x$ and $\sigma_{j,2}\beta_i \bmod x$: he simply gives u and v.

4 Non-Interactive Perfect Zero-Knowledge for Threshold Statement

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

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

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

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 non residue 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 $(x, \vec{y}) \in \mathrm{T}(k, m)$, 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 $(x, \vec{y}) \in \mathrm{T}(k, m)$. Moreover, the proof is non-interactive (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)

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 $\mathcal{B}(x, y, w, b)$ in the following way:

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

We say that the prover (x, y)-*opens* w as b if he proves that $\mathcal{B}(x, y, w, b) = 1$.
If $y \in QR_x$ then $\mathcal{B}(x, y, w, 0) = \mathcal{B}(x, y, w, 1) = 1$ 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 $\mathcal{B}(x, y, w, 0) = 1$ (and thus the prover can (x, y)-open w as a 0) and $\mathcal{B}(x, y, w, 1) = 0$, as, by the fact that -1 is a quadratic non residue modulo x, $-w \bmod x$ is a quadratic non residue modulo x. Now, suppose that $w \in NQR_x$. Then $\mathcal{B}(x, y, w, 1) = 1$, as, by the fact that -1 is a quadratic non residue modulo x, $-w \bmod x$ is a quadratic residue modulo x, (and thus the prover can (x, y)-open w as a 1) and obviously $\mathcal{B}(x, y, w, 0) = 0$. 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, $\mathcal{B}(x, y, w, b)$ is proven to hold by running C on input $(x, (-1)^{1-b} w \bmod x, -y \bmod x)$.

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 P 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-uple $(S_1, \ldots, S_m)$ represents a (k, m)-sequence of admissible shares for b. Now, why is this a proof that less than k elements of $\vec{y}$ are quadratic non residues?
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 $|I| \geq k$. 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 $1/|\mathcal{E}| \leq 1/m$. 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 Figure 1. Here (C,D) is a non-interactive perfect ZK proof system for OR. Our field $\mathcal{E}$ is the field with $2^{\lceil \log(m+1) \rceil}$ elements.

Theorem 6. (P,V) is a Non-Interactive Perfect Zero-Knowledge Proof System for the language $\mathrm{T}(k, m)$.

Proof. Omitted.

Remark. The protocol (P,V) can be easily extended to a proof system for the language of pairs $(x, \vec{y})$, where the number of quadratic non residues in $\vec{y}$ is greater or equal to k, that is:

$$\overline{\mathrm{T}}(k,m) = \{\, (x, \vec{y}) \,|\, x \in \mathrm{BL},\ y_i \in Z_x^{+1},\ \text{for } i = 1, \ldots, m,\ \text{and } |\{y_i | y_i \in NQR_x\}| \geq k \,\}.$$

The prover uses the algorithm P on input $(x, -y_1 \bmod x, \ldots, -y_m \bmod x)$.

5 Non-Interactive Perfect Zero-Knowledge for General Access Structures Statements

In general a secret sharing scheme is a procedure to share a secret among a certain number of participants so that only qualified subsets of participants can reconstruct the secret. Threshold schemes are particular secret sharing schemes where the set of qualified subsets consists of all the subsets with at least k participants. The set of qualified subsets is called access structure. For obvious reasons an access structure $\mathcal{A}$ has to be monotone; i.e., if $A \in \mathcal{A}$ then all A' that contain A also belong to $\mathcal{A}$.

As done for threshold scheme, to each access structure $\mathcal{A}$, we associate a language $\mathrm{AS}(\mathcal{A})$ of pairs $(x, \vec{y})$, where x is a Blum integer and $\vec{y}$ is an m-uple of elements of Z_x^{+1}, in the following way. For each access structure $\mathcal{A} = \{A_1, \cdots, A_k\}$, where $A_i = \{a_{i1}, \cdots, a_{ik_i}\} \subseteq \{0, 1, \cdots, m\}$, we can define a predicate $p_{\mathcal{A}}(x, \vec{y})$ as follows:

$$p_{\mathcal{A}}(x, \vec{y}) = \begin{cases} 1 & \text{if for each } i = 1, \ldots, k, \text{ at least one out of } \{y_{i1}, \cdots, y_{ik_i}\} \\ & \text{is a quadratic residue modulo } x \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Then the language $\mathrm{AS}(\mathcal{A})$ is the language of pairs $(x, \vec{y})$ for which $p_{\mathcal{A}}(x, \vec{y}) = 1$.

We consider polynomial-time ideal secret sharing schemes, that is schemes in which the secret and its shares are taken from the same set (for instance $\mathrm{GF}(q)$, where q is a prime), the algorithm of the dealer is polynomial-time, and it is possible to verify in polynomial time that a given set of shares reconstructs the

Input to P and V:

- A reference string η.
- $(x, \vec{y}) \in \mathrm{T}(k, m)$, where $|x| = n$ and $\vec{y} = (y_1, \ldots, y_m)$.

(Set $\eta = \tau \circ \sigma$, where $\sigma = b \circ \rho \circ \gamma_{11} \circ \cdots \circ \gamma_{1m} \circ \cdots \circ \gamma_{\lceil \log(m+1) \rceil 1} \circ \cdots \circ \gamma_{\lceil \log(m+1) \rceil m}$, $|b| = 1$ and ρ is the concatenation of $\rho_{ij} \in Z_x^{+1}$, $1 \leq i \leq \lceil \log(m+1) \rceil$ and $1 \leq j \leq m$.)

Instructions for P.

P.1 (*Prove that x is a Blum integer.*)

Run the algorithm A on input x using the random string τ and send its output *Pf*.

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 $s_{1j} \ldots s_{\lceil \log(m+1) \rceil j}$.

Randomly choose S_l, with l such that $y_l \in QR_x$, in such a way that $(S_1, \ldots, S_m)$ constitutes a (k, m)-sequence of admissible shares for the bit b.

For l such that $y_l \in QR_x$,
  let $s_{1l} \ldots s_{\lceil \log(m+1) \rceil l}$ be the binary representation of S_l.

P.3 (*Prove that the sequence of admissible shares has been correctly constructed.*)

For $i = 1, \ldots, \lceil \log(m+1) \rceil$,
  for $j = 1, \ldots, m$,
    (x, y_j)-open ρ_{ij} as s_{ij} by running the algorithm 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}.

Input to V:

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

Instructions for V.

V.1 (*Verify that x is a Blum integer.*)

Run the algorithm B on input x and τ thus verifying *Pf*.

V.2 (*Verify the admissibility of the sequence of shares.*)

Verify that the m-uple $(S_1, \ldots, S_m)$ 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. 1. The proof system (P,V) for $\mathrm{T}(k, m)$.

secret. Supposing the existence of such a secret sharing scheme, a non-interactive perfect ZK proof system for AS($\mathcal{A}$) can be obtained in the following way. Let $(x, \vec{y}) \in$ AS($\mathcal{A}$) be the input to the protocol. Similarly to the protocol (P,V), the prover picks from the reference string $m\lceil\log(m+1)\rceil$ 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-uple $(S_1, \ldots, S_m)$ represents a secret sharing scheme for b for the access structure $\mathcal{A}$ on $\mathcal{P}$.

Theorem 7. Let $\mathcal{P}$ be a set of m participants and $\mathcal{A}$ a monotone access structure on it. Suppose there exist a polynomial-time secret sharing scheme for the access structure $\mathcal{A}$. Then the protocol described above is a non-interactive perfect zero-knowledge proof system for AS($\mathcal{A}$).

Proof's sketch: Using the existence of a polynomial-time ideal secret sharing scheme for the access structure $\mathcal{A}$, and ideas similar to those of protocol (P,V), one can see that the theorem holds. □

Notice that in the above theorem no restriction is imposed upon the size of $\mathcal{A} = \{A_1, \cdots, A_k\}$; e.g., k can be exponential in m (the length of the vector $\vec{y}$). This is actually what happens in threshold schemes.

References

1. M. Bellare and M. Yung, *Certifying Cryptographic Tools: The case of Trapdoor Permutations*, in CRYPTO 92.
2. M. Ben-Or, O. Goldreich, S. Goldwasser, J. Hastad, S. Micali, and P. Rogaway, *Everything Provable is Provable in Zero Knowledge*, in "Advances in Cryptology – CRYPTO 88", vol. 403 of "Lecture Notes in Computer Science", Springer Verlag, pp. 37–56.
3. G. R. Blackley, *Safeguarding Chryptographic Keys*, Proceedings AFIPS 1979 National Computer Conference, pp. 313–317, June 1979.
4. J. Boyar, K. Friedl, and C. Lund, *Practical Zero-Knowledge Proofs: Giving Hints and Using Deficiencies*, Journal of Cryptology, n. 4, pp. 185–206, 1991.
5. M. Blum, A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge*, SIAM Journal of Computing, vol. 20, no. 6, Dec 1991, pp. 1084–1118.
6. M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and Applications*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 103–112.
7. R. Boppana, J. Hastad, and S. Zachos, *Does co-NP has Short Interactive Proofs ?*, Inf. Proc. Lett., vol. 25, May 1987, pp. 127–132.
8. A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge Proof-Systems*, in "Advances in Cryptology – CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag, pp. 52–72.
9. A. De Santis, G. Persiano, and M. Yung, *Perfect Zero-Knowledge Proofs for Graph Isomorphism Languages*, manuscript.

10. U. Feige, D. Lapidot, and A. Shamir, *Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String*, in Proceedings of 22nd Annual Symposium on the Theory of Computing, 1990, pp. 308–317.
11. L. Fortnow, *The Complexity of Perfect Zero-Knowledge*, Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, pp. 204–209.
12. O. Goldreich and E. Kushilevitz, *A Perfect Zero Knowledge Proof for a Decision Problem Equivalent to Discrete Logarithm*, in "Advances in Cryptology - CRYPTO 88", Ed. S. Goldwasser, vol. 403 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 57–70.
13. O. Goldreich, S. Micali, and A. Wigderson, *Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Design*, Proceedings of 27th Annual Symposium on Foundations of Computer Science, 1986, pp. 174–187.
14. O. Goldreich, S. Micali, and A. Wigderson, *How to Play Any Mental Game*, Proceedings of the 19th Annual ACM Symposium on Theory of Computing, pp. 218–229.
15. S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989.
16. R. Impagliazzo and M. Yung, *Direct Minimum Knowledge Computations* "Advances in Cryptology – CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag pp. 40–51.
17. I. Niven and H. S. Zuckerman, *An Introduction to the Theory of Numbers*, John Wiley and Sons, 1960, New York.
18. A. Shamir, *How to share a secret*, Communication of the ACM, vol. 22, n. 11, November 1979, pp. 612–613.
19. G. J. Simmons, *An Introduction to Shared Secret and/or Shared Control Schemes and Their Application*, Contemporary Cryptology, IEEE Press, pp. 441–497, 1991.
20. D. R. Stinson, *An Explication of Secret Sharing Schemes*, Design, Codes and Cryptography, Vol. 2, pp. 357–390, 1992.
21. M. Tompa and H. Woll, *Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information*, Proc. 28th Symposium on Foundations of Computer Science, 1987, pp. 472–482.