Public-Randomness in Public-Key Cryptography

EXTENDED ABSTRACT

Alfredo De Santis*
Dipartimento di Informatica ed Applicazioni
Università di Salerno
84081 Baronissi (Salerno), Italy

Giuseppe Persiano†
Aiken Comp. Lab.
Harvard University
Cambridge, MA 02138

Abstract

In this work we investigate the power of Public Randomness in the context of Public-key cryptosystems. We consider the Diffie-Hellman Public-key model in which an additional short random string is shared by all users. This, which we call Public-Key Public-Randomness (PKPR) model, is very powerful as we show that it supports simple non-interactive implementations of important cryptographic primitives.

We give the first *completely* non-interactive implementation of Oblivious Transfer. Our implementation is also secure against receivers with unlimited computational power.

We propose the *first* implementation of non-interactive nature for Perfect Zero-Knowledge in the dual model of Brassard, Crépeau, and Chaum for all NP-languages.

1 Introduction

The Public Key model, introduced by Diffie and Hellman [DH], suggests an elegant and efficient way to eliminate the need for preliminary secure interaction which is essential in Private-Key Cryptography. Each party A publishes in a Public File his encryption key P_A and keeps secret his decryption key S_A. Once the public file has been established, each user can receive any number of encrypted messages on a public channel from any other user that has access to the public file, without having to interact with them via exchanges of messages. Moreover, [DH] showed how to

*Part of this work was done while the author was visiting IBM Research Division, T. J. Watson Research Ctr, Yorktown Heights, NY 10598.

†Partially supported by ONR Grant #N00039-88-C-0163.

I.B. Damgard (Ed.): Advances in Cryptology - EUROCRYPT '90, LNCS 473, pp. 46-62, 1991.

produce unforgeable signatures for messages and how two users could establish a private key in this (non-interactive) model.

The security of the protocols rests upon the very natural assumption that only limited computational resources are available to each user. The introduction of Complexity considerations in Cryptography caused much excitement and made possible applications never thought of before. Most notably, the *Oblivious Transfer* (OT) protocol by Rabin (see [HR]) and the *Zero-Knowledge Proof System* of [GMR] and [GMW1]. However, all proposed implementations of the OT require the ability to interact and therefore cannot be used in the Public-Key model. Different non-interactive or bounded-interaction scenarios in which Zero-Knowledge was possible have been proposed ([BFM], [DMP1], [DMP2], [DMP3], [BDMP], [K], [KMO]) but they all suffer of some practical drawbacks that limited their applicability.

In this paper, we consider the Diffie-Hellman model in which a short random string is shared beforehand by all users. We call this model Public-Key Public-Randomness Cryptosystem (PKPR Cryptosystem). The set up of the PKPR Cryptosystem does not require any preprocessing stage: Each user chooses and validates by himself his own public and private keys without any interaction. Moreover, no center or distributed fault-tolerant computation (in the sense of [GMW2, BGW, CCD, GHY, B, RB]) is ever invoked to protect against possible "cheating". Even though interaction is never allowed, the PKPR Cryptosystem is very powerful as we show that important cryptographic primitives have simple non-interactive implementations in this model.

Summary of the results. We give a *completely non-interactive* implementation of Oblivious Transfer in the PKPR model. This is the first non-interactive implementation of OT that does not require a trusted center or some distributed fault-tolerant computation.
Our implementation is essentially optimal. Indeed, a recent result of Ostrowsky and Yung [OY] shows that it is not possible to achieve a non-interactive OT from scratch. We prove that a PKPR setting is enough to achieve non-interactive OT.
Our implementation is also secure against receivers with unlimited computing power. We give the *first* implementation of non-interactive nature for Perfect Zero-Knowledge (in the dual model of [BC] and [Ch]) for all NP-languages. Unlike previous implementations of non-interactive Zero Knowledge with a common random string, our implementation is very simple and, most notably, allows any number of provers to be active (this solves the open problem of *many independent provers* (see [DMP1]), though in a slightly modified scenario).

Our results are based on the well known and widely used Quadratic Residuosity Assumption, and they demonstrate the added value of short Public Randomness in the context of Public key Cryptography.

2 Preliminaries

In this section we review some elementary facts from number theory about quadratic residues and the probabilistic encryption scheme based on the difficulty of deciding quadratic residuosity of [GM]. We follow the notation of [BDMP].

For each natural number x, the set of positive integers less than x and relatively prime to x form a group under multiplication modulo x denoted by Z_x^*. $y \in Z_x^*$ 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 non residue* modulo x. The *quadratic residuosity predicate* is defined as follows

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

If $y_1, y_2 \in Z_x^*$, then

1. $Q_x(y_1) = Q_x(y_2) = 0 \Longrightarrow Q_x(y_1 y_2) = 0$.
2. $Q_x(y_1) \neq Q_x(y_2) \Longrightarrow Q_x(y_1 y_2) = 1$.

For any fixed $y \in Z_x^*$, the elements $\{yq \bmod x \mid q \text{ is a quadratic residue modulo } x\}$ constitute an equivalence class that has the same cardinality as the class of quadratic residues.
The set Z_x^{+1} is the subset of Z_x^* consisting of all elements with Jacobi symbol $+1$.
The problem of deciding quadratic residuosity consists of evaluating the predicate Q_x. This is easy when the modulus x is prime and appears to be hard when is composite. Indeed, no efficient algorithm is known for deciding quadratic residuosity modulo composite numbers whose factorization is not given. Actually, the fastest way known consists of first factoring x and then compute $Q_x(y)$. This fact has been first used in cryptography by Goldwasser and Micali [GoMi]. We use it in this paper with respect to the following special moduli.

Blum integers. Let $n \in \mathcal{N}$. The set of Blum integers of size n, $BL(n)$, is defined as follows: $x \in BL(n)$ if and only if $x = pq$, where p and q are primes of length n both $\equiv 3 \bmod 4$. These integers were introduced in Cryptography by M. Blum [Bl]. Blum integers are easy to generate. There exists an efficient algorithm that, on input 1^n, outputs the factorization of a randomly selected $x \in BL(n)$. This class of integers constitutes the hardest input for any known efficient factoring algorithm. Since no efficient algorithm is known for deciding quadratic residuosity modulo Blum integers, this justifies the following

Quadratic Residuosity Assumption (QRA): For each efficient poly-size family of circuits $\{C_n\}_{n \in \mathcal{N}}$, all positive constants d, and all sufficiently large n,

$$Pr\Big(x \leftarrow BL(n);\ y \leftarrow Z_x^{+1} : C_n(x,y) = Q_x(y)\Big) < 1/2 + n^{-d}.$$

That is, no poly-size family of circuits can guess the value of the quadratic residuosity predicate substantially better than by random guessing. This assumption has been

used for the first time by [GM] and is now widely used in Cryptography. For instance, the proof system of [BDMP] is based on it.

2.1 Encryption schemes

In a seminal paper, Goldwasser and Micali [GM] introduced a Public-key encryption scheme whose security is based on the quadratic residuosity assumption. The public key of user B contains a random integer x_B product of two primes of the same length and y_B, a random quadratic non residue modulo x_B with Jacobi symbol $+1$. B's secret key contains the prime factors of x_B. To secretly send B an l-bit message $M = m_1 \cdots m_l$, A just encrypts each bit m_i by computing $c_i = y_B^{m_i} r_i^2 \bmod x_B$, where r_i is a randomly chosen element of $Z_{x_B}^*$. It is easily seen that each c_i corresponding to a 0 bit is a quadratic residue and each c_i corresponding to a 1 bit is a quadratic non residue modulo x_B. This observation gives a very simple decryption algorithm: it is enough for B to compute the quadratic residuosity of the c_i's he has received.

What makes decryption possible in this scheme is the fact that y_B is a quadratic non residue. In fact, should y_B be a quadratic residue, then, independently of the value of m_i, each c_i is a quadratic residue, and therefore B has absolutely no *information* to recover the bits m_i's.

In what follows, we will denote by $E(x, y, m)$ the algorithm that returns a random encryption of m computed using the pair (x, y) and $D(p, q, c)$ the algorithms that returns the decryption of the ciphertext computed using x's prime factors p and q.

3 Public-Key Public-Randomness Cryptosystems

A *Public-Key Public-Randomness (PKPR) Cryptosystem* consists of:

(a) a random string σ, called the *reference* string;

(b) a set of transactions $\mathcal{T}$;

(c) a key-space K_σ, which is the set of keys associated with σ; (each key is a pair $< PK, SK >$, where PK is the *public key* and SK is the *private key*;)

(d) a probabilistic polynomial time algorithm *Key_Generator* which returns randomly chosen elements in K_σ;

(e) a poly-time algorithm *Verify_PublicKey* outputting VALID/NONVALID;

(f) for each $T \in \mathcal{T}$ a send/receive pair of probabilistic polynomial time algorithms (S_T, R_T).

Initialization of the PKPR cryptosystem.
Each user U chooses by himself his own public and private keys by following the procedure:

- Randomly select a key $< PK_U, SK_U >$ by running *Key-Generator* on input σ.
- PK_U is made public. The private key SK_U is kept secret.

Performing a transaction.
Let $T \in \mathcal{T}$ be a transaction (defined by a protocol program, e.g. oblivious transfer, secure message sending, Zero-Knowledge proof, etc.). User A, having x as input, performs T with another user B, by following the procedure:

- Verify B's public key PK_B by making sure that *Verify_PublicKey* (σ, PK_B) =VALID.
- Run the (possibly probabilistic) procedure $S_T(PK_B, x)$ and send B the output y.

B does not reply upon receiving y, he just privately computes $R_T(y, PK_A, SK_B)$.

Notice that each transaction is non-interactive. Also, for the set-up of the Cryptosystem no center or distributed fault-tolerant computation is required: each user chooses and validates by himself his own public key. Any other user can check the correctness of the construction of a public key, without any interaction. All is needed is the availability of a common random reference string, whose length does not depend on the number of users, but only on the desired level of security.

4 Oblivious Transfer

Oblivious Transfer has been introduced by Rabin (see [HR]), who first gave an implementation (for honest players) based on the difficulty of factoring. OT is a protocol for two parties: the *Sender* who has a string s, and the *Receiver*. Each of the following two events is equally likely to occur at the end of the protocol.

- The *Receiver* learns the string s.
- The *Receiver* does not get any information about s.

Moreover, at the end of the protocol, the *Sender* does not know whether the *Receiver* got s or not. The wide applicability of the OT was recognized since the early days of modern cryptography; the paper by Blum [B2] is a an example of how OT can be used to implement several other protocols.

A different flavor of OT, the *1-out-2 Oblivious Transfer* was later introduced by Even, Goldreich, and Lempel [EGL]. Here, the sender has two strings s_0 and s_1. Each of the following two events is equally likely to occur at the end of a 1-out-2 Oblivious Transfer:

- The *Receiver* learns the string s_0, and does not get any information about s_1.
- The *Receiver* learns the string s_1, and does not get any information about s_0.

The sender has no information on which string the receiver gets. It is clear that 1-out-2 Oblivious Transfer can be used to implement an Oblivious Transfer. Crépeau [Cr] showed how to achieve a 1-out-2 Oblivious Transfer by using Rabin's OT, thus establishing their equivalence. The rest of this section is organized as follows. In Section 4.1 we formally define what we mean by Non-Interactive Oblivious Transfer (NIOT) in the PKPR model and in Section 4.2 we give an implementation of NIOT in the PKPR model.

4.1 Non-Interactive Oblivious Transfer in the PKPR model

A Non-Interactive Oblivious Transfer is a quadruple of algorithms *(Key_Generator, Verify_PublicKey, Sender, Receiver)*. Suppose the sender A has two strings (s_0, s_1) that he wants to obliviously transfer to the receiver B. Informally, the mechanics of the transfer is the following. First, A verifies B's public file PK_B by making sure that $Verify\_PublicKey(\sigma, PK_B)$ =VALID. Then A computes and sends B the message $msg = Sender(PK_B, s_0, s_1)$. To retrieve one of the two strings, B computes $Receiver(PK_B, SK_B, \sigma, msg)$, where SK_B is its own private key.

Definition 1 A Non-Interactive Oblivious Transfer is a quadruple of algorithms *(Key_Generator, Verify_PublicKey, Sender, Receiver)*, where *Key_Generator* and *Sender* are probabilistic polynomial time and *Verify_PublicKey* and *Receiver* are deterministic polynomial time, such that

1. *Meaningfullness: The receiver gets one of the two strings.*
2. *Verifiability: The validity of the construction can be efficiently verified.*
3. *1 out-of 2: The receiver gets only one string and not even a bit of the other.*
4. *Obliviousness: The sender cannot predict which string is going to be received.*

4.2 Implementing Oblivious Transfer in the PKPR Model

We show how to implement a (non-interactive) Oblivious Transfer in the PKPR setting. At this aim, we describe the *Key_Generator* algorithm and the *Verify_PublicKey* algorithm needed to initialize the public file and the *Sender* and *Receiver* algorithm to actually perform the OT.

Algorithm *Key_Generator*(σ)
Input: An n^3-bit reference string $\sigma = \sigma_1 \circ \ldots \circ \sigma_{n^2}$, where $|\sigma_i| = n$ for $i = 1, 2, \ldots, n^2$.

1. *Select public and secret keys.*
   Randomly select two n-bit primes $p, q \equiv 3 \bmod 4$ and set $x = pq$.
   Randomly select $r \in Z_x^*$, $z \in Z_x^{+1}$ and compute $y = -r^2 \bmod x$.
2. *Validate public key.*
   Set Val= empty string.
   For $i = 1, \ldots, n^2$
   if $\sigma_i \in Z_x^{+1}$ then
   if $Q_x(\sigma_i) = 0$ then append $\sqrt{\sigma_i} \bmod x$ to Val.
   if $Q_x(\sigma_i) = 1$ then append $\sqrt{y\sigma_i} \bmod x$ to Val.
   else append σ_i to Val.
3. Set $PK = (x, y, z, Val)$ and $SK = (p, q)$.

Output:(PK, SK).

Algorithm *Verify_PublicKey*(σ, PK)
Input: An n^3-bit reference string $\sigma = \sigma_1 \circ \ldots \circ \sigma_{n^2}$, where $|\sigma_i| = n$ for $i = 1, 2, \ldots, n^2$. A Public Key $PK = (x, y, z, Val)$, where $Val = v_1 \circ \ldots \circ v_{n^2}$.

For $i = 1, \ldots, n^2$
if $\sigma_i \in Z_x^{+1}$ then verify that either $\sigma_i = v_i^2 \bmod x$ or $y\sigma_i = v_i^2 \bmod x$.

Output: If all checks are successfully passed then output VALID else NONVALID.

The validation of the public key is just a non-interactive Zero-Knowledge proof of the NP statement "x is product of two primes and y is a quadratic non residue modulo x". This proof is obtained in a direct manner, that is without making use of reduction to $3SAT$ and, most importantly, the same string σ can be used by any number of users to certify their own public key entry.

To prove that the algorithm *Verify_PublicKey* satisfies the *Verifiability* requirement, we have to show that if either x is not product of two primes or y is not a quadratic non residue then with very high probability the algorithm *Verify_PublicKey* outputs NONVALID. As this is a non-interactive Zero-Knowledge proof, the proof of the verifiability can be obtained by using the techniques of [DMP1] and [BDMP].

Algorithm *Sender*(PK, s_0, s_1)
Input: A public key $PK = (x, y, z, Val)$. Two strings s_0, s_1.

Compute and send the pair $msg = (E(x, z, s_0), E(x, zy \bmod x, s_1))$.

Algorithm *Receiver*(PK, SK, σ, msg)
Input: A public key $PK = (x, y, z, Val)$ along with the corresponding secret key $SK = (p, q)$. An n^3-bit reference string $\sigma = \sigma_1 \circ \ldots \circ \sigma_{n^2}$, where $|\sigma_i| = n$ for $i = 1, 2, \ldots, n^2$. A pair $msg = (\alpha, \beta)$.

1. If z is a quadratic residue then
   If $D(\alpha, p, q) = 0$ then output $D(\beta, p, q)$ else STOP.
2. If z is a quadratic non residue then
   If $D(\beta, p, q) = 0$ then output $D(\alpha, p, q)$ else STOP.

Theorem 1 *Under the QRA, the above quadruple of algorithms (Key_Generator, Verify_PublicKey, Sender, Receiver) is a Non-Interactive Oblivious Transfer.*

Proof. We shall prove that the quadruple (*Key_Generator, Verify_PublicKey, Sender, Receiver*) meets the four requirements of Definition 1. We have already seen that the *Verifiability* requirement is met. Thus, all is left to prove is that *Meaningfullness*, *1 out-of 2*, and *Obliviousness* are met too.

Meaningfullness and *1 out-of 2.*
As y_B is a quadratic non residue and x_B is product of two primes, for each $z \in Z^{+1}_{x_B}$ exactly one between z_B and $y_B z_B \bmod x_B$ is a quadratic non residue. Therefore, B receives exactly one of s_0, s_1. If B receives s_0 (s_1), then s_1 (s_0) has been encrypted using a quadratic residue modulo x and all B gets is a sequence of random quadratic residues modulo x_B.

Obliviousness.
We prove that the existence of a pair of algorithms (ADV_0, ADV_1) such that for some $c > 0$, all sufficiently large n, and all (s_0, s_1)

$$Pr\,(\sigma \leftarrow \{0,1\}^n\,;\ (PK, SK) \leftarrow \mathit{Key\_Generator}(\sigma);\ msg \leftarrow ADV_0(\sigma, PK, s_0, s_1)\ :\ \mathit{Receiver}(SK, PK, \sigma, msg) \neq \text{STOP} \wedge ADV_1(PK, \sigma, s_0, s_1, msg) = \mathit{Receiver}(SK, PK, \sigma, msg)\ ) \geq 1/2 + n^{-c}$$

contradicts the QRA. We shall in fact exhibit an algorithm $Q(\cdot, \cdot)$ that decides quadratic residuosity that uses ADV_0 and ADV_1 as subroutines.

Algorithm $Q(x,z)$.

Input: $x \in BL(n)$, $z \in Z_x^{+1}$

1. *Construct Public key*
   Set σ and *Val*=empty string.
   Randomly select $r \in Z_x^*$ and set $y = -r^2 \bmod x$.
   For $i = 1, \ldots, n^2$
      Randomly select an n-bit integer s_i.
      If $s_i \notin Z_x^{+1}$ append s_i to σ.
        else
          Toss a fair coin
          If HEAD then append $ys_i^2 \bmod x$ to σ and ys_i to *Val*.
          If TAIL then append $s_i^2 \bmod x$ to σ and s_i to *Val*.
2. Set $PK = (x, y, z, Val)$ and select two strings s_0, s_1.
3. Set $(\alpha, \beta) = ADV_0(\sigma, PK, s_0, s_1)$
4. If $ADV_1(PK, \sigma, s_0, s_1, (\alpha, \beta)) = s_0$ then **Output**(1) else **Output**(0).

Let us now compute

$$Pr(x \leftarrow BL(n); z \leftarrow Z_x^{+1} : Q(x,z) = Q_x(z)) = 1/2\,(Pr(x \leftarrow BL(n); z \leftarrow QNR_x : Q(x,z) = 1) + Pr(x \leftarrow BL(n); z \leftarrow QR_x : Q(x,z) = 0))$$

where QR_x and QNR_x are the classes of quadratic residues and quadratic non residues modulo x, respectively.

Now, we observe that if $z \in QRN_x$ then certainly $Receiver(SK, PK, \sigma, msg) = s_0$, as s_0 has been encrypted using z. Therefore

$$Pr(x \leftarrow BL(n); z \leftarrow QNR_x^{+1} : Q(x,z) = 1) \geq 1/2 + n^{-c}$$

The same reasoning can be done for the case in which $z \in QR_x$, thus yielding

$$Pr(x \leftarrow BL(n); z \leftarrow Z_x^{+1} : Q(x,z) = Q_x(z)) \geq 1/2 + n^{-c}$$

which contradicts the QRA.

QED

Dependent Oblivious Transfers.

The same public key, $PK_B = (y_B, z_B)$, can be used to perform many OT's. However, these OT's will not be independent as the outcome of one of them determines the outcome of all of them. This is useful when we want that only one of two strings is received and not even one single bit of the other is leaked. If j independent OT's are

desired, it is enough to have j different $z_B \in Z^{+1}_{x_B}$ in B's public key. The problem of obtaining j independent OT's using the same public key (whose size does not depend on j) has its own interest and is currently open. In the final paper we give a partial solution to it based on k-wise independence.

1-out-k Oblivious Transfer.
An immediate generalization of the 1-out-2 OT is the 1-out-k OT. This is a protocol where A transfer to B exactly one of k strings $s_0, s_1, ..., s_{k-1}$, in such a way that (1) B does not get any information on the other $k-1$ strings, and (2) A does not know which string B got. Our protocol can be easily modified to implement a 1-out-k OT. User B randomly chooses a Blum integer x_B, and $z^0_B, ..., z^{k-1}_B \in Z^{+1}_{x_B}$, such that exactly one z^i_B is a quadratic non residue. Then, B computes $PK_B = (x_B, z^0_B, ..., z^{k-1}_B, Val_B)$, SK_B = (prime factors of x_B), where Val_B = (Non-Interactive Zero-Knowledge proof, in the sense of [BFM], that PK_B has been correctly computed). To oblivious transfer one of $s_0, s_1, ..., s_{k-1}$, user A computes and sends B a random permutation π of them, computes and sends B the encryptions $E(x_B, z^i_B, s_{\pi(i)})$, $i = 0, ..., k-1$. Notice that if A does not permute the s_i this is a protocol for 1-out-k disclosure in which the receiver "secretly" chooses 1 out of k secrets (defined in [BCR]).

5 Non-Interactive Perfect Zero-Knowledge Arguments

Bit-Commitment in the Public-Key Public-Randomness Model.
A bit commitment protocol is a fundamental 2-party cryptographic protocol. It allows one party A to hide (commit) one bit b from the other party B and, later, to show (decommit) it to B. Even though B does not know which bit A has committed to, he is guaranteed that the bit decommitted is the bit A originally committed to. If B is poly-time, the bit commitment can be implemented as: A chooses a secure encryption scheme (in the sense of [GM]) and commits to a bit by encrypting it. B, being poly-time, cannot decrypt and compute b. To decommit b, A simply shows the random bit used for the encryption.
Suppose that A is polynomial time and B has infinite computing power. We use the unconditionally secure blobs of [BC] for the commitment of A to B. B publishes in his public file a random integer product of two primes x_B, s_B a random quadratic residue modulo x_B, and a NIZK proof of correctness of the publication. To commit to a bit b, A performs the procedure: (1) randomly select $r \in Z^*_{x_B}$; (2) compute and send B $w = s^b_B r^2 \bmod x_B$. If A wants to decommit the bit b committed by w, he reveals r, that must be the square of either w or $w/s_B \bmod x_B$. It is clear that B, by just looking at w, cannot figure out whether A knows a root of w or $w/s_B \bmod x_B$. On the other hand, if A knows a square root of both w and $w/s_B \bmod x_B$, then he can compute a square root of s_B as well. This implies that A can extract square roots of random squares and thus he is able to factor random composite numbers, in

contradiction with the Quadratic Residuosity Assumption. Notice that the same pair (x_B, s_B) can be used to perform any number of bit commitments to B. Later, we will use this bit commitment scheme to obtain Non-Interactive Perfect Zero-Knowledge Arguments for all NP languages.

Zero-Knowledge Proofs and Arguments.
Goldwasser, Micali, and Rackoff [GMR] introduced the notion of Interactive Zero-Knowledge Proof System. This allows an infinitely powerful (but not trusted) prover to convince a probabilistic polynomial time verifier that a certain theorem is true without revealing any other information. They also distinguished between *Computational Zero-Knowledge* and *Perfect Zero-Knowledge*. In a Computational Zero-Knowledge Proof all the information about the theorem and its proof is given, but it would take more than polynomial time for the verifier to extract it. On the other hand, a Perfect Zero-Knowledge Proof conveys no information at all and even an all powerful verifier could not extract any information from it. Perfect Zero-Knowledgeness is thus a desirable property in a proof as a prover can never be sure of the computational power of the verifier he is talking to. Unfortunately, although all NP languages have Computational Zero-Knowledge proofs [GMW1], Perfect Zero-Knowledge for an NP-complete language would cause the Polynomial Time Hierarchy to collapse [F, BHZ].

The concept of *Interactive Zero-Knowledge Argument* has been introduced by Brassard and Crépeau [BC] and Chaum [Ch]. Here, we have a probabilistic polynomial time prover that wants to convince a (possibly unlimited computing powerful) verifier that a certain theorem is true in Zero-Knowledge. (The term "Argument" is used instead of "Proof" as an all powerful prover could cheat the verifier.) As the verifier may have infinite computational power, in this setting all Zero-Knowledge Arguments must indeed be Perfect Zero-Knowledge. Surprisingly, Brassard and Crépeau [BC] were able to show that, in this scenario, all NP languages have Perfect Zero-Knowledge Arguments.

Although, different non-interactive scenarios in which Zero-Knowledge Proofs are possible have been proposed, all implementations of Perfect Zero-Knowledge Arguments require interaction. A protocol requiring only constant number of rounds is due to Brassard, Crépeau, and Yung [BCY].

We show that *Non-Interactive Zero-Knowledge Arguments* for all NP languages are possible in the PKPR setting, under the Quadratic Residuosity Assumption. Moreover, our implementation supports any polynomial (in the length of the reference string σ) number of distinct provers. Unlike [BFM], [DMP1], [BDMP], our proofs are directed to a single user, and thus are not publicly verifiable.

To prove that all NP languages have Perfect Zero-Knowledge Arguments, it is enough to prove it for an NP-complete language. We choose Hamiltonian graphs as NP-complete language. The ingredients for our construction are Blum's interactive protocol [B3], an efficient technique of [KMO], our implementations of OT and bit commitment schemes. More efficient protocols can be obtained by suitably using the interactive protocol of [IY] that simulates directly any given computation, or envelope-based protocols as in [KMO].

Non-Interactive Perfect Zero-Knowledge Arguments for Hamiltonian Graphs. We show how to give a Perfect Zero-Knowledge Argument that a graph is Hamiltonian, in the Public-Key Public-Randomness Model. This in turn yields Perfect Zero-Knowledge Argument for all NP languages. We first describe the content of the public file and then show how the proof is actually performed.

The public file. V randomly chooses two primes p_V and q_V of the same length, computes $x_V = p_V q_V$, then V chooses s_V a random quadratic residue modulo x_V, y_V, a random quadratic non residue modulo x_V, and a random $z_V^1, ..., z_V^n \in Z_{x_V}^{+1}$. V publishes the public key $PK_V = (x_V, s_V, y_V, z_V^1, \ldots, z_V^n, Val_V)$, where $Val_V =$ (NIZK proof that PK_V has been correctly computed), and keeps secret the private key $SK_V = (p_V, q_V)$. The NIZK proof is computed on input the reference string and certifies that x_V is a Blum integer and s_V (y_V) is a quadratic residue (non residue) modulo x_V.

Proving that G is Hamiltonian. Suppose that the prover P wants to show that the graph G has an Hamiltonian path to a (possibly unlimited computing powerful) verifier V. He performs the following program.

For $i = 1, \ldots, n$ do

1. Randomly select a permutation π_i of the vertices of G and compute $\hat{G}_i = \pi_i(G)$, an isomorphic copy of G.
2. Using s_V, commit bit by bit the adjacency matrix of $\hat{G}_i$ and the permutation π_i.
3. Let α_i be the concatenation of the decommitment keys of the adjacency matrix of $\hat{G}_i$ and the permutation π_i and β_i the concatenation of the decommitment keys of the entry of the adjacency matrix of $\hat{G}_i$ that corresponds to edges in the Hamiltonian path.

   Oblivious transfer the pair (α_i, β_i) using y_V^i and z_V^i.

We now give an informal proof that the above protocol is a Perfect Zero-Knowledge Argument. A formal proof will be given in the final version. For each i, the verifier V gets either the adjacency matrix of $\hat{G}_i$ and the permutation π_i (and thus can check that $\hat{G}_i$ is indeed an isomorphic copy of G) or an hamiltonian path in $\hat{G}_i$ (and thus can check that $\hat{G}_i$ is Hamiltonian). By the properties of the Oblivious Transfer, P does not know which one V is going to receive. Therefore, if G is not Hamiltonian, either $\hat{G}$ is not isomorphic to G (in which case P is caught with probability 1/2) or $\hat{G}$ is not Hamiltonian (again P is caught with probability 1/2). Thus, under the QRA, there is only an exponentially low probability for a poly-time prover to convince a verifier that a non-hamiltonian graph is hamiltonian (soundness requirement of a proof system). The Perfect Zero-Knowledgeness of the protocol follows from the properties of the commitment scheme and of the NIZK proof contained in Val_V.

6 Related works

Non-Interactive Oblivious Transfer. Recently, Ostrowsky and Yung [OY] proved that it is not possible to achieve a non-interactive OT from scratch. That is, two parties that have never met beforehand need at least 2 rounds to perform an OT. It is then natural to ask: What additional resources are required for a "non-interactive OT" to be possible? We prove that a PKPR setting is enough to achieve non-interactive OT. Adopting a different perspective, we prove that 2 rounds are enough for an OT from A to B, if a random string is available. The rounds are (1) B selects and sends A his public key, and (2) A performs a non-interactive OT using B's public key. Thus, our implementation is essentially optimal in view of [OY].

Non-interactive Oblivious Transfer with a Center. Bellare and Micali [BM] introduced the problem of achieving non-interactive OT in a Public-key scenario where a trusted center is on duty and proposed an implementation based on a complexity assumption related to the Discrete Logarithm problem (the Diffie-Hellman assumption [DH]). In their model, the trusted center publishes a central public key and each user that wants to put his key on the public file has to validate it by interacting with the center. A center may be replaced by a collective distributed fault-tolerant computation if the majority of users is "honest". We show that a short random string can replace the trusted center (or a distributed fault-tolerant computation). Therefore, in our model, there is no need for interaction to validate the public file: each user can prove the correctness of the choice of his own public key by himself.

Non-interactive Zero-Knowledge. Blum, Feldman, and Micali [BFM] (with improvements in [DMP1] and [BDMP]) were the first to propose a non-interactive scenario in which Zero-Knowledge (in the sense of [GMR]) proofs of membership for all NP-languages were possible. A random short reference string is required for the proof-system. Their implementations can only support a limited number of provers. Recently, De Santis and Yung [DY] and Feige, Lapidot, and Shamir [FLS] proved that many provers can share the same random string. All these implementations are quite involved. Other models in which non-interactive (or bounded-interaction) Zero-Knowledge is achievable have been proposed. [DMP2] proved that after a preprocessing stage, it is possible to give a NIZK proof of any short NP-theorem. This proof-system is based on a very general assumption (existence of one-way functions) but has limited applicability. [K] showed how to achieve Zero-Knowledge proofs using only OT's. [KMO] gave protocols more efficient than [K], and furthermore showed that it is possible to move all the needed OT's in a short pre-processing stage. All these protocols are restricted to two parties.

Non-interactive Zero-Knowledge with a Center. Bellare and Micali [BM] showed how to implement NIZK proofs in a Public-key cryptosystem in which a center is available. In their implementation, the center has to compute and publish a central public key (which is common to all users). Then, each user chooses by himself its own public

key, and must validate it by proving the correctness of its computation by shortly interacting with the center. Their implementation is based on a complexity assumption related to the Discrete Logarithm problem (the Diffie-Hellman assumption [DH]), whereas our implementation is based on the Quadratic Residuosity Assumption.

The Public-key Public-Randomness model shows how to dispose of an active center by having just a short random string known beforehand to all users. To validate his choice of the public key, each user computes by himself and publishes a proof of correctness in his public file.

Perfect Zero-Knowledge Arguments. Brassard and Crépeau [BC] and Chaum [Ch] consider a model where the prover is poly-bounded and the verifier has unlimited computational power. All NP-languages have Perfect Zero-Knowledge Arguments (that is Zero-Knowledge proofs in the BCC model). Actually Brassard, Crépeau, and Yung [BCY] proved that there are perfect Zero-Knowledge arguments that use a constant number of rounds (only 6) for any NP-language. In this paper, we prove that all NP-languages have Non-Interactive Perfect Zero-Knowledge Arguments in the PKPR setting. Our is the first implementation of non-interactive nature for Perfect Zero-Knowledge Arguments.

Note added in proof

Bert van Boer [Bo] has independently proved that it is possible to have a non-interactive Oblivious Transfer after an interactive preprocessing stage. His protocol relies on the Quadratic Residuosity Assumption. Using the techniques we discussed in this paper, his implementation can be adapted to work also in the PKPR setting.

7 Acknowledgments

We are very grateful to Moti Yung for encouragement and helpful discussions.

References

[B] D. Beaver, *Secure Multiparty Protocols Tolerating Half Faulty Processors*, CRYPTO 1989.

[Bl1] M. Blum, *Coin Flipping by Telephone*, IEEE COMPCON 1982, pp. 133–137.

[Bl2] M. Blum, *Three Applications of the Oblivious Transfer*, Unpublished manuscript.

[Bl3] M. Blum, *How to Prove a Theorem So No One Else Can Claim It*, Proceedings of the International Congress of Mathematicians, Berkeley, California, 1986, pp. 1444–1451.

[BC] G. Brassard and C. Crépeau, *Non-transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond*, Proceedings of the 27th IEEE Symp. on Foundation of Computer Science, 1986, pp. 188–195.

[BCC] G. Brassard, C. Crépeau, and D. Chaum, *Minimum Disclosure Proofs of Knowledge*, Journal of Computer and System Sciences, vol. 37, no. 2, October 1988, pp. 156–189.

[BCR] G. Brassard, C. Crépeau, and J.-M. Robert, *Information Theoretic Reductions among Disclosure Problems*, Proceedings of the 27th IEEE Symp. on Foundation of Computer Science, 1986, pp. 168–173.

[BCY] G. Brassard, C. Crépeau, and M. Yung, *Everything in NP can be Proven in Perfect Zero-Knowledge in a Bounded Number of Rounds*, ICALP 89.

[BDMP] M. Blum, A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero Knowledge*, MIT Research Report MIT/LCS/TM-430, May 1990.

[BFM] M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge Proof Systems and Applications*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, 1988.

[BGW] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computations*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 1–10.

[BHZ] R. Boppana, J. Hastad, and S. Zachos, *Does co-NP have Short Interactive Proofs?*, Information Processing Letters, vol. 25, May 1987, pp. 127–132.

[BM] M. Bellare and S. Micali, *Non-interactive Oblivious Transfer and Applications*, CRYPTO 1989.

[Bo] B. van Boer, *Oblivious Transfer Protecting Secrecy*, Eurocrypt 90.

[Ch] D. Chaum, *Demonstrating that a Public Predicate can be Satisfied Without Revealing any Information About How*, in "Advances in Cryptology – CRYPTO 86", vol. 263 of "Lecture Notes in Computer Science", Springer Verlag, pp. 195–199.

[Cr] C. Crépeau, *Equivalence Between Two Flavors of Oblivious Transfer*, in "Advances in Cryptology - CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag, pp. 350–354.

[CCD] D. Chaum, C. Crépeau, and I. Damgård, *Multiparty Unconditionally Secure Protocols*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, 1988, pp. 11–19.

[DH] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, vol. IT-22, no. 6, Nov. 1976, pp. 644–654.

[DMP1] 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.

[DMP2] A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge Proof-Systems with Preprocessing*, in "Advances in Cryptology - CRYPTO 88", Ed. S. Goldwasser, vol. 403 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 269–282.

[DMP3] A. De Santis, S. Micali, and G. Persiano, *Removing Interaction from Zero-Knowledge Proofs*, in "Advanced International Workshop on Sequences", Positano, Italy, June 1988, Ed. R. M. Capocelli, Springer-Verlag, pp. 377–393.

[DY] A. De Santis and M. Yung, *Cryptographic Applications of Metaproofs*, CRYPTO 90.

[EGL] S. Even, O. Goldreich, and A. Lempel, *A Randomized Protocol for Signing Contracts*, CACM, vol. 28, 1985, pp. 637–647.

[F] L. Fortnow, *The Complexity of Perfect Zero-Knowledge*, Proceedings 19th Annual ACM Symposium on Theory of Computing, New York, 1987, pp. 204–209.

[FLS] U. Feige, D. Lapidot, and A. Shamir, *Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String*, FOCS 90.

[GHY] Z. Galil, S. Haber, and M. Yung, *Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model*, in "Advances in Cryptology – CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag, pp. 135–155.

[GM] S. Goldwasser and S. Micali, *Probabilistic Encryption*, Journal of Computer and System Science, vol. 28, n. 2, 1984, pp. 270–299.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989.

[GMW1] 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.

[GMW2] O. Goldreich, S. Micali, and A. Wigderson, *How to Play Any Mental Game*, Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York, 1987, pp. 218–229.

[HR] J. Halpern and M. O. Rabin, *A Logic to Reason about Likelihood*, Proceedings of the 15th Annual Symposium on the Theory of Computing, 1983, pp. 310–319.

[IY] R. Impagliazzo and M. Yung, *Direct Minimum Knowledge Computations*, in "Advances in Cryptology – CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag pp. 40–51.

[K] J. Kilian, *Founding Cryptography on Oblivious Transfer*, Proceedings 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, 1988, pp. 20–31.

[KMO] J. Kilian, S. Micali, and R. Ostrowsky, *Minimum-Resource Zero-Knowledge Proofs*, Proceedings of the 30th IEEE Symposium on Foundation of Computer Science, 1989.

[OY] R. Ostrowsky and M. Yung, *On Necessary Conditions for Secure Distributed Computation*, preprint 1989.

[RB] T. Rabin and M. Ben-Or, *Verifiable Secret Sharing and Multiparty Protocols with Honest Majority*, Proceedings of the 21st Annual ACM Symposium on Theory of Computing, Seattle, Washington, 1989, pp. 73–85.