Appeared in *Crypto87*, Springer Verlag, Lecture Note in Computer Science (293), pages 73–86.
Reproduced (in June 1997) from an old troff file.

How to Solve any Protocol Problem – An Efficiency Improvement

(Extended Abstract)

Oded Goldreich and *Ronen Vainish*

Department of Computer Science
Technion – Israel Institute of Technology, Haifa, ISRAEL.

ABSTRACT: Consider n parties having local inputs $x_1, x_2, ..., x_n$ respectively, and wishing to compute the value $f(x_1, ..., x_n)$, where f is a predetermined function. Loosely speaking, an n-party protocol for this purpose has *maximum privacy* if whatever a subset of the users can efficiently compute when participating in the protocol, they can also compute from their local inputs and the value $f(x_1, ..., x_n)$. Recently, Goldreich, Micali and Wigderson have presented a polynomial-time algorithm that, given a Turing machine for computing the function f, outputs an n-party protocol with maximum privacy for distributively computing $f(x_1, ..., x_n)$. The maximum privacy protocol output uses as a subprotocol a maximum privacy two-party protocol for computing a particular simple function $p_1(\cdot, \cdot)$. More recently, Haber and Micali have improved the efficiency of the above n-party protocols, using a maximum privacy two-party protocol for computing another particular function $p_2(\cdot, \cdot)$. Both works use a *general* result of Yao in order to implement protocols for the *particular* functions p_1 and p_2. In this paper, we present direct solutions to the above two particular protocol problems, avoiding the use of Yao's general result. In fact, we present two alternative approaches for solving both problems. The first approach consists of a simple reduction of these two problems to a variant of *Oblivious Transfer*. The second approach consists of designing direct solutions to these two problems, assuming the intractability of the Quadratic Residuosity problem. Both approaches yield simpler and more efficient solutions than the ones obtained by Yao's result.

This research was supported by grant No. 86-00301 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

Updated affiliation for Oded Goldreich – Department of Computer Science and Applied Mathematics Weizmann Institute of Science, Rehovot, ISRAEL. Email: `oded@wisdom.weizmann.ac.il`

Preface (by O.G., June 1997): This write-up was reproduced (without substantial proofreading) from an old `troff` file which contained the *Crypto87* version of this work. The conversion was done automatically using some software (which is not perfectly fit for the input file). Thus, I've superficially gone over the output and eliminated some failures of the conversion (but I may not have caught them all). I did not try to improve the exposition.

In fact, I was amazed at the poor level of the exposition. It requires a drastic improvement which I'm not going to perform now. In particular, the overall organization is not spelled-out: So I've added an organization paragraph at the end of the Introduction.

Finally, I'd like to call the readers attention to the simple protocol presented in Subsection 3.1. This protocol utilizes Oblivious Transfer in order to present a maximum-privacy 2-party protocol for a specific Boolean function on three bits. Private computation of this specific function is all that is required for the multi-party maximum-privacy protocol generator of [GMW2] (in the version attributed to [HM]).

1. INTRODUCTION

The main purpose of many cryptographic protocols is to allow parties to collaborate towards some common goal, while maintaining the maximum possible privacy of their secrets. Typically, the common goal is to compute some function of the local inputs (secrets) held by the different parties. Maximum privacy means that this value is distributively computed without revealing more about the local inputs than what is revealed by the value itself. More formally, let x_i be the local input of party i ($1 \leq i \leq n$), and f be an n-argument function. The parties wish to obtain the value $f(x_1, ..., x_n)$, but do not wish to leak any further information about their local inputs. To better understand what is meant by this requirement, consider the situation when all parties trust an additional party. In this case, each party may (secretly) send his local input to the *trusted party*, which will then compute the value of the function, and announce this value to all parties. A maximum privacy protocol achieves the effect of the trusted party without using a trusted party. Namely, whatever a party can efficiently compute when participating in a maximum privacy protocol, he could have efficiently computed after participating in the above "trusted party" protocol. To better understand what is meant by maximum privacy, consider the problem of computing the sum of the local inputs (i.e. $f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} x_i$). A maximum privacy protocol for computing the sum of the local inputs guarantees that whatever a coalition T of parties can efficiently compute when participating in the protocol, can also be efficiently computed from their local inputs ($\{x_i\}_{i \in T}$) and the sum of all local inputs (i.e. $\sum_{i=1}^{n} x_i$). Equivalently, all they learned about the local inputs of the other parties is their sum (i.e. $\sum_{i \notin T} x_i$), and this of course can not be avoided. Goldreich, Micali and Wigderson [GMW2] have proposed a method for generating maximum privacy protocols for computing any function f. Their method is in fact a polynomial-time algorithm that given as input a Turing Machine description of the (n-argument) function f, outputs a maximum privacy n-party protocol for computing f. These protocols use instances of a maximum privacy two-party subprotocol for the following particular protocol problem (in S_5 – the symmetric group of 5 elements):

Input: A's local input is a permutation, $\tau \in S_5$, while B's local input is a permutation $\sigma \in S_5$.
Output: A's local output is a permutation $\tau' \in S_5$, and B's local output is a permutation $\sigma' \in S_5$ such that $\tau \cdot \sigma = \sigma' \cdot \tau'$. (Here $\cdot$ means permutation composition.)

Recently, an efficiency improvement of the [GMW2] algorithm has been suggested by Haber and Micali [HM]. The protocols output by their algorithm use instances of a maximum privacy two-party subprotocol for the following particular problem (in $GF(2)$ arithmetic):

Input: A's local input is a pair of bits a_1 and a_2, while B's input consists of the bits b_1 and b_2.
Output: A's local output is a bit a_0, while B's local output is a bit b_0 such that $a_0 + b_0 = \sum_{i=1}^{2} a_i \cdot b_i$.

Both, [GMW2] and [HM] use for solving the above problems, a general result of Yao [Y2]: a method for generating maximum privacy two-party protocols for any two-argument function. This method (modified in [GMW2] by using ideas from [EGL]) guarantees maximum privacy under the assumption that trapdoor one-way permutations exist. In this paper, we present direct solutions to the above two particular protocol problems, avoiding the use of Yao's general result. In fact, we present two alternative approaches for solving both problems. The first approach consists of a simple reduction of these two problems to a variant of *Oblivious Transfer*, which can implemented assuming the existence of any trapdoor one-way permutation. The second approach consists of designing direct solutions to these two problems, assuming the intractability of the Quadratic Residuosity problem. Both approaches yield simpler and more efficient solutions than the ones obtained by Yao's result. Our protocols and their applications are presented in a model where parties follow the protocol properly, except that they may store all intermediate computations done during the execution. Thus, we concentrate only in guaranteeing that the protocols have maximum privacy. Using the results of [GMW2], each maximum privacy protocol in the above model can be transformed into a protocol guaranteeing both maximum privacy and correctness of output in a model where a minority of the parties may deviate from the protocol in arbitrary (but polynomial-time) manner.

Organization (added in June 1997):

Section 2 (Preliminaries): This section contains general background material (Subsections 2.1–2.4) as well as specific background (Subsections 2.5–2.7).

Section 3 (Solving the GF(2) scalar-product problem): This is the more important part of the work and the reason I've borthered at all to reproduce it from a troff file. It contains a simple solution using 1-out-of-2 Oblivious Transfer and zero-knowledge proofs (Subsection 3.1) as well as a more efficient solution based on Quadratic Residousity (rest of the section). The bare solution of Subsection 3.1 (without zero-knowledge proofs) is efficient but only holds for the "semi-honest" model. The solution presented in Subsection 3.2 (and analyzed in Subsect. 3.3–3.5) holds for a more general model in which the adversary is allowed arbitrary polynomial-time behaviour which does not violates the correctness of the computation. This model is called the *value-preserving adversary model* (see Subsection 2.4).

Section 4 (Solving the S_5 permutation problem): The ideas of Section 3 are extended to deal with a more complex problem. There is no real reason to care about this part (since the problem of Section 3 is sufficient for the envisioned application).

2. PRELIMINARIES

In this section, we recall the basic definitions and notations used in this paper.

2.1. Two-party Cryptographic Protocols

Loosely speaking, a two-party cryptographic protocol is a pair of programs run by a corresponding pair of interacting Turing Machines. An *interactive Turing Machine* is a Turing machine with the following tapes:

1) A read only *input tape*.

2) A read only *random tape*.

3) A read/write *working tape*.

4) A pair of communication tapes, one being read-only and the other write-only.

5) A write only *output tape*.

The machine can be thought of as using the bits of the random tape as coin tosses, sending messages through its write-only communication tape, and receiving messages on its read-only communication tape. It should be noted that the current configuration of the machine is determined by the context of its input tape, random tape, and read-only communication tape. Two interactive machines A and B, are called an *interactive pair* (of Turing machines) if they share their communication tapes in the obvious manner (i.e. A's read-only communication tape is B's write-only communication tape and via versa). We consider only polynomial-time protocols. These are protocols consisting of pairs of programs, such that the running time of each program is polynomial in the length of the input. Typically, we will not be interested in a particular execution of a protocol, but rather in the probability distribution on the set of possible executions. This probability distribution is a function of the local inputs and internal coin tosses of the interacting machines. A particularly intersting probability distribution is defined by a party's view of the execution. The party's view of an execution contains the contents of his local input and random tapes, as well as the contents of his read-only communication tape. We stress that the contents of the input tape and the read-only communication tape can not be modified (or erased) by the program. We denote by $A_{B(y)}(x)$ the probability distribution defined by A's view of an execution, in which A has local input x, and B has local input y.

2.2. Polynomial Indistinguishability

A fundamental notion regarding probability distributions is the inability to efficiently tell them apart. This notion is captured by the definition of polynomially indistinguishable probability ensembles originating in [GM, Y1] and sketched below. A probability ensemble $Y = (Y_1, Y_2, ...)$ is an infinite sequence of probability distributions, where Y_k is a probability distribution on binary strings. Typically, the support of Y_k will contain strings of length polynomial in k. A *test*, T, is a probabilistic polynomial time algorithm that on input a string x output a bit b. Let $P^T_{Y_k}$ denote the probability that T outputs 1 on input a string randomly selected with a probability distribution Y_k. Two ensembles X and X' are *polynomially indistinguishable* if for all tests T, for all constants $c > 0$, and for sufficiently large k,

$$|P^T_{X_k} - P^T_{X'_k}| < \frac{1}{k^c}$$

2.3. The Privacy Requirement

In the introduction, we have motivated maximum privacy protocols as ones allowing the distributed computation of functions without revealing more about the local inputs than what is revealed by the value of the function. It was required that whatever a party can efficiently compute when participating in a maximum privacy protocol, he could have efficiently computed from his local input and the value of the function. Clearly, it suffices to require that he can efficiently compute his view of the execution from his local input and the value of the function. A formal definition follows. (For first reading of the definition, assume that z is the integer k in unary representation and that its sole purpose is to allow the used of the formalism of polynomial indistinguishability.)

Definition 1 (A program preserves privacy with respect to a particular program.): Let Π be a probability ensemble $(\Pi_1, \Pi_2, ...)$ such that Π_k is a probability distribution on triples (x, y, z). Program B *preserves the privacy of f with respect to program A* if there is a probabilistic polynomial-time machine M, that for every ensemble Π, when given input x, z and $f(x, y)$ outputs $M(x, z, f(x, y))$ such that ensemble $M(x, z, f(x, y))$ is polynomially indistinguishable from the ensemble $A_{B(y)}(x, z)$. (Here the triple (x, y, z) is chosen with probability distribution Π_k.)

In fact, we allow z to be arbitrary, thus capturing a priori information that party A might have had on the input of B. This way, we guarantee that even with the help of such a priori information, executing the protocol does not reveal more about the local inputs than is revealed by the value of $f(x, y)$. Although maximum privacy is defined here with respect to the computation of functions, the definition naturally extends to the computation of probability distributions.

2.4 Two Models of Party's Behavior

In this paper we consider two types of party's behavior. The first type, called *semi-honest behavior* consists of a party following his program while recording all intermediate computing steps on a special tape (called the *history tape*) and conducting an arbitrary polynomial-time computation using the history tape as an input. Note that even if a program of a semi-honest specifies that it has to erase the contents of his working tape, this contents still appear on the history tape. Yao [Y2] (resp. Goldreich, Micali and Wigderson [GMW2]) presents a method of "forcing" the participants of any two-party (resp. multi-party) protocol to behave in a semi-honest manner.

Definition 2 (Protocol which preserves privacy in the semi-honest model): A two-party protocol (A,B) *preserves the privacy of f in the semi honest model* if program A preserves privacy with respect to program B and program B preserves privacy with respect to program A.

Definition 3 (Maximum privacy protocol.): A protocol (A, B) has *maximum privacy* if program A preserves privacy with respect to all polynomial-time program B^* and program B preserves privacy with respect to all polynomial-time program A^*.

When talking about a cryptographic protocol we are usually interesting in two properties, correctness and privacy (correctness means that the true value of f is being computing by the protocol.). In this paper we are concerned only with the privacy condition. This can be motivated in two ways. First we believe that privacy and correctness are distinct notions which are better understood when dealt separately. Correctness is easily dealt using zero-knowledge proofs [GMW1], while privacy even in the semi-honest model requires different techniques [GMW2]. Secondly, it is natural to consider setting in which the parties are very interested in obtaining the correct value of the function and on top of this seek to gain additional information (but not at the cost of not getting the correct value). This is formulated by the following behavior model. A *value-preserving adversary*, consists of a party which may deviates from the protocol in any manner that does not change the true value of f. We introduce two protocols for the same problem, the first preserve privacy in the semi-honest model while the second has maximum privacy.

2.5 One-out-of-Two Oblivious Transfer

Susan and Ron are friends. Susan has two secret bit, which Ron wants. In order to preserve their friendship Susan is willing to give Ron only one of her secret at his choice, but Ron does not want her to know which secret he chose. A *one-out-of-two Oblivious Transfer*, denoted OT_2^1, is a two-party protocol which guarantees that Ron gets only the secret (bit) he has chosen while Susan does not know which secret he chose. The OT_2^1 as motivated above, must be related to a model of behavior. We consider OT_2^1 in the semi-honest model and in the value-preserving adversary model. When we say that a protocol implements OT_2^1 in a specific model we mean that the OT_2^1 properties hold when the party's behavior is restricted to is model.

2.6. The Quadratic Residuosity Problem

Let m be a composite integer, the product of two large primes p and q. We denote by Z_m^* the multiplicative group modulo m. The set of quadratic residue modulo m is denoted by

$$Q_m = \{a : \exists x \in Z_m^* \text{ s.t. } a \equiv x^2(m)\}$$

For every $a \in Z_m^*$, the Jacobi symbol of a mod m, denoted $(\frac{a}{m})$, is defined as $(\frac{a}{p}) \cdot (\frac{a}{q})$, where $(\frac{a}{p})$ is $+1$ if a is a quadratic residue modulo p and -1 otherwise. The Jacobi symbol $(\frac{a}{m})$ can be easily computed from a and m. Clearly, $(\frac{a}{m}) = -1$ implies $a \notin Q_m$, but the converse does not hold. In fact, distinguishing elements of Q_m from quadratic non-residues mod m (with Jacobi Symbol 1) is considered intractable. (This computation is easy if the factorization of m is known.) To concentrate on elements with Jacobi Symbol 1, we denote

$$Z_m^{(+1)} = \{a \in Z_m^* : (\tfrac{a}{m}) = +1\}.$$
$$N_m = Z_m^{(+1)} - Q_m$$

The *Quadratic Character* of x mod m, denoted $QC_m(x)$, is defined as 0 if $x \in Q_m$ and 1 otherwise. The *Quadratic Residuosity problem* is to determine, on input x and m, the value of $QC_m(x)$. This task is considered intractable in the following sense

Intractability Assumption of Quadratic Residuosity [GM]: Let $C = \{C_i\}$ be an infinite sequence of Boolean circuits such that C_i has $2i$ input bits. Let f_{C_i} denote the fraction of integers m product of two primes, of length $i/2$ bits each, such that for every $x \in Z_m^{(+1)}$, $C_i(x,m) = QC_m(x)$. Then, for every family of polynomial size circuits, $C = \{C_i\}$, every constant $c > 0$ and sufficiently large i, $f_{C_i} < i^{-c}$. (Here the size of a circuit family $C = \{C_i\}$ is a function mapping i to the number of gates in C_i.)

We use the fact that, under the intractability assumption of Quadratic Residuosity, it is infeasible to guess the quadratic character with any non-negligible advantage over $1/2$.

Definition: We say that C *polynomially approximates* Quadratic Residuosity, if there exist a constant $c > 0$ such that for infinitely many i's

$$Prob\,(C_i(x,m) = QC_m(x)) > \tfrac{1}{2} + \tfrac{1}{n^c}$$

where the probability is taken over all possible $m = p \cdot q$ (with p and q being two primes of length $i/2$ each) and all $x \in Z_m^{(+1)}$ with uniform probability distribution.

Theorem [GM]: Under the intracability Assumption of Quadratic Residuosity, there exist no family of polynomial size circuits that polynomially approximates Quadratic Residuosity.

2.7. Notations

- Let S be a finite set. By $e \in_R S$ we mean an element randomly chosen from the set S with uniform probability distribution.
- We denote by S_5 the group formed by the set of all permutations over {1,2,3,4,5}, and permutation composition as operator. (This group is known as the symmetric group.)

3. THE GF(2) SCALAR PRODUCT PROTOCOL

In this section we present a maximum privacy two-party protocol for the problem of distributively computing scalar product in $GF(2)$, defined as follows:

Input: A's local input is a t-dimensional binary vector $\bar{a} = (a_1, a_2, ..., a_t)$, while B's input is another t-dimensional binary vector $\bar{b} = (b_1, b_2, ..., b_t)$.

Output: A's local output is a bit a_0, while B's local output is a bit b_0 such that $a_0 + b_0 = \sum_{i=1}^{t} a_i \cdot b_i$.

In fact, we are interested in the case $t = 2$, which is exactly the subprotocol required for the Haber Micali protocol generator [HM]. For simplicity, we present a protocol for the following related problem:

Input: A's local input is a pair of bits a_0 and a_1, while B's input is a single bit b_1.

Output: B's local output is a bit b_0 which equals $a_0 + a_1 \cdot b_1$. (A has no local output.)

It is easy to reduce the original problem to the later problem. Alternatively, one may use the ideas of the protocol described below to directly solve the original problem.

3.1 Protocol For Semi-Honest Using OT_2^1.

A defines its first secret to be a_0 and his second secret to be $a_0 + a_1$. Using OT_2^1 B chooses one of A's secrets according to the value of b_1. If $b_1 = 0$ then B chooses the first secret, otherwise he chooses the second secret. It easy to see that B's output bit equals $a_0 + a_1 b_1$, thus correctness holds. The privacy in the semi-honest model holds by the definition of OT_2^1 in the semi-honest model. An OT_2^1 is simply implemented in the semi-honest model, assuming the existence of trapdoor one way permutation [GMW2], unfortunately this implementation does not have maximum privacy. Zero knowledge proofs can be used in order to ensure that this OT_2^1 protocol has maximum privacy, however the modified protocol is no longer simple and efficient. An efficiency improvement have been achieved in the next maximum privacy protocol for the scalar product problem, that is under the Quadratic Residuosity Assumption.

3.2. The Protocol in the Value-Preserving Adversary model.

Preprocessing: B chooses at random two k-bit primes p, q. (k is the security parameter)
B computes $m = p \cdot q$. Next, B chooses $y \in_R N_m$ and publishes the couple m, y.

i) B chooses $s \in_R Z_m^*$ and computes $\beta = (s^2 \cdot y^{b_1} \bmod m)$. B sends β to A.

ii) A chooses $r \in_R Z_m^*$ and computes $\alpha = (r^2 \cdot y^{a_0} \cdot \beta^{a_1} \bmod m)$. A sends α to B.

iii) B checks the quadratic residuosity of α, and sets $b_0 = QC_m(\alpha)$.

3.3. Correctness of the Protocol

We first show, that the above protocol is correct; namely that the output satisfies the specification conditions.

Claim 1: The bit b_0 computed by B does equal $a_0 + a_1 \cdot b_1$.

Proof: B gets

$$\alpha \equiv r^2 \cdot y^{a_0} \cdot \beta^{a_1} \equiv r^2 \cdot y^{a_0}(s^2 \cdot y^{b_1})^{a_1} \equiv (r \cdot s^{a_1})^2 \cdot y^{a_0 + a_1 b_1} \pmod{\mathrm{m}}.$$

B set $b_0 = QC_m(\alpha) \equiv a_0 + a_1 \cdot b_1 \pmod 2$. $\square$

3.4. Maximum Privacy of the Protocol

We now prove that the above protocol has the maximum privacy property. First we use the Intractability Assumption of subsection 2.6 to prove that B preserves privacy with respect to any A^*, and next we prove that A preserves privacy with respect to any B^* (using no assumptions). By Definition 1, program B preserves privacy with respect to A^* if there exists a machine which, on input the local inputs of A^* and the value of the function, simulates the interaction between A^* and B. This requirement has to be satisfied for any possible input that A^* may have, including encoding of possible a-priori information on B's inputs (denoted z). However, if the modulus m is chosen in the preprocessing and is input to the protocol then z may depend on it. In particular, z may contain the prime factorization of m and in such a case clearly B does not preserve privacy. This problem may be resolved in one of the following ways:

1) Having B choose m at random each time the protocol is executed, instead of having it chosen in a preprocessing stage. This completely solves the problem, at the cost of substantially decreasing the efficiency of the protocol.

2) Leaving the protocol as it is, and relaxing the definition of privacy preserving. The definition is relaxed by restricting z to be polynomial-time computable. In particular, $z = R(b_1, m, y)$, where R is a probabilistic polynomial-time algorithm. (Thus, z may be a random variable.) This restriction is justified by the applications of the above protocol. Typically, the protocol will be used many times, each time with the same modulus (m) but with possibly different a_0, a_1, b_1. When considering the i-th application of the protocol, the input to A^* is the history of the previous $i-1$ applications. One can then use induction on i to show that privacy is preseved in i successive applications of the protocol. In the induction step, we use the fact that the history of the previous $i-1$ can be simulated by a probabilistic polynomial-time machine, and thus the input z in the current application satisfies the restriction.

In the following Claim, we use adopt the second alternative. The reader may easily modify our proof to show that the modified protocol (as suggested in the first alternative) preserves privacy in the original sense (of Definition 1).

Claim 2: Assuming intractability of Quadratic Residuosity and restricting z to be polynomial-time computable (see (2) above), program B of the above protocol preserves privacy with respect to any A^*.

Proof's Sketch: To prove the claim, we demonstrate a machine M which on input a_0, a_1, m, y and z (as restricted above) outputs a probability distribution $M(a_0, a_1, m, y, z)$ which is polynomially indistinguishable from $A^*_{B(b_1,p,q,m,y)}(a_0, a_1, m, y, z)$. Machine M proceeds as follows:

(Simulates step (i) of machine B): Sets $b'_1 = 1$, chooses $s \in_R Z^*_m$, and computes $\beta' = (s^2 \cdot y^{b'_1} \bmod m)$. Outputs its inputs together with β' and stops.

We will show that the output of M is polynomially indistinguishable from the contents of the input and read-only communication tapes of A^* (when interacting with B). The only potential

difference between $M(a_0, a_1, m, y, z)$ and $A^*_{B(b_1,p,q,m,y)}(a_0, a_1, m, y, z)$ may be created by a difference between the distribution of β and β'.
There are essentially two cases.
Case 1: $Prob(b_1 = 1) > 1 - k^{-c}$, for all $c > 0$ and sufficiently large k. In such a case, the ensembles $M(\cdots)$ and $A_{B(\cdots)}(\cdots)$ are almost the same and can not be polynomially distinguished (regardless of the difficulty of determining Quadratic Residuosity).
Case 2: There exist a constant $c > 0$ such that $Prob(b_1 = 0) > k^{-c}$ for infinitely many k's. Assume, on the contrary, that there is a (polynomial-time) test T distinguishing $M(a_0, a_1, m, y, z)$ from $A^*_{B(b_1,p,q,m,y)}(a_0, a_1, m, y, z)$, when (a_0, a_1, m, y), (b_1, p, q, m, y) and z are taken from a distribution, denoted Π_k, in which p, q are randomly selected k-bit primes, $m = pq$, $y \in_R N_m$, and $z = R(b_1, m, y)$, where R is a probabilistic polynomial-time machine. In such a case, we use the test T to construct a family of circuits for approximating Quadratic Residuosity. (Details follow.)
Let I_k be a value of (a_0, a_1) for which the test T distinguishes the above two ensembles. With no loss of generality, assume that T outputs 1 with higher probability on the ensemble $M(\cdots)$ than on the ensemble $A_{B(\cdots)}(\cdots)$.
The k-th circuit incorporates I_k and the test T, working as follows: On input a k-bit composite m and $x \in Z_m^{(+1)}$, the circuit computes y and z as explained below, feeds T with (I_k, m, y, z, x) and outputs T's answer. (x is placed in the position of β (β').) It is left to specify the computation of y and z.
The circuit chooses $y \in_R Z_m^{(+1)}$, and computes $z = R(b_1, m, y)$ using the probabilistic polynomial-time machine R (which is incorporated in the circuit). Note that the test T will determine correctly the quadratic character of $x \in Z_m^{(+1)}$, in case $y \in N_m$. We do not know how T behaves in case $y \in Q_m$. Therefore, before using y to test the quadratic character of x we estimate the behavior of the test with this y. Namely, we select many $r_i \in_R Q_m$ (by letting $r_i = s_i^2$, where $s_i \in_R Z_m^*$) and feed the test T with either (I_k, m, y, z, r_i) or $(I_k, m, y, z, r_i \cdot y)$. If the test T distinguishes these two cases, we use this y for determining the quadratic character of x (i.e. feed T with (I_k, m, y, z, x)). Otherwise, we try again.
One can show that the circuits constructed as above do approximate Quadratic Residuosity with a non-negligible advantage. The technical details are quite standard, and are omitted here. We reach a contradiction to the Quadratic Residuosity Assumption, and the Claim follows. □ We now prove preservation of privacy with respect to B. This time we use no intractability assumptions.

Claim 3: The protocol preserves privacy with respect to B.
Proof's Sketch: We will show that there exist a machine M which on input b_1, k, z and $f(a_0, a_1, b_1)$ ($= a_1 \cdot b_1 + a_0 \bmod 2$), outputs a probability distribution which is identical to the distribution on B's input and read-only tapes during interaction with A. M operates as follows.

1) M randomly chooses two k-bit primes p, q, computes $m = p \cdot q$, and chooses $y \in_R (Z_m^{(+1)} - Q_m)$.

2) M chooses $r' \in_R Z_m^*$, computes $\alpha' \equiv r'^2 \cdot y^{f(a_0,a_1,b_1)} \pmod m$, and outputs (its input and) α'.

Recall that A calculate α as $\alpha \equiv r^2 \cdot y^{a_0} \cdot \beta^{a_1} \equiv (r \cdot s_1^a)^2 \cdot y^{a_0+a_1 b_1} \pmod{m}$, where $r \in_R Z_m^*$. Since both $r \cdot s^{a_1}$ and r' are uniformly distributed in Z_m^*, α and α' have identical probability distribution. The Claim follows. $\square$

3.5. Summing Up

Combining Claims 1, 2 and 3, we get

Theorem 1: The above protocol is a maximum privacy protocol for the simplified $GF(2)$ scalar product problem.

A protocol for the original $GF(2)$ scalar product problem, can be easily derived and proven using the above ideas. The protocol of subsection 3.1 is modified as follows. In step (i), B chooses $s_1, s_2, ..., s_t \in_R Z_m^*$, computes $\beta_i = (s_i^2 \cdot y^{b_i} \bmod m)$, and sends $\beta_1, \beta_2, ..., \beta_t$ to A. In step (ii), A computes $\alpha_i = \beta_i^{a_i}$, chooses $r \in_R Z_m^*$ and $a_0 \in_R \{0,1\}$, computes $\alpha = (r^2 \cdot y^{a_0} \cdot \Pi_{i=1}^t \alpha_i \bmod m)$, and sends α to B. In step (iii), B sets $b_0 = QC_m(\alpha)$ $(=a_0 + \sum_{i=1}^t a_i b_i)$.

4. THE PERMUTATION SWITCHING PROTOCOL

In this section, we present a two-party protocol with maximum degree privacy for the problem of switching permutations defined as follows:
Input: A's local input is a permutation, $\tau \in S_5$, while B's local input is a permutation $\sigma \in S_5$.
Output: A's local output is a permutation $\tau' \in S_5$, and B's local output is a permutation $\sigma' \in S_5$ such that $\tau \cdot \sigma = \sigma' \cdot \tau'$.

An equivalent formulation used in the sequel is
Input: A's local input is a pair of permutations, $\tau, \tau' \in S_5$. B's local input is a permutation $\sigma \in S_5$.
Output: B's local output is $\sigma' = \tau \cdot \sigma \cdot \tau'$. ($A$ has no local output.)

One-out-of 120 Oblivious Transfer can be implements to solve the above problem in the semi-honest model. A maximum privacy protocol for the permutation switching problem is following presents, that is under the Quadratic Residuosity Assumption.

4.0. Conventions

Throughout this section, we use quite non-standard representation of permutations. The reason for this representation is that it allows to composite a non-encrypted permutation with an encrypted permutation resulting in an encrypted permutation. We represent permutations in S_5 by quintuples of distinct elements in {1,2,3,4,5}. By $(i_1, i_2 ... i_5)$ we mean the permutation mapping i_k to k ($\forall k = 1...5$). For example let $\sigma = (3,5,1,2,4)$, then $\sigma \cdot (A,B,C,D,E) = (C,E,A,B,D)$. Assume that we encrypt quintuples $(i_1, ..., i_5)$ by encrypting each element separately; namely $E(i_1, ..., i_5) = E(i_1), ..., E(i_5)$. Then, given $\sigma = (3,5,1,2,4)$ and $E(i_1, ..., i_5)$ we can compute

$$E(\sigma \cdot (i_1, i_2, i_3, i_4, i_5)) = E(i_3, i_5, i_1, i_2, i_4) = E(i_3), E(i_5), E(i_1), E(i_2), E(i_4) = \sigma \cdot E(i_1, ..., i_5)$$

The representation used above allows us to compute $E(\tau \cdot \sigma)$ from τ and $E(\sigma)$. We wish to be able to compute $E(\tau \cdot \sigma)$ from $E(\tau)$ and σ. Using two particular encryption formats, E and $\tilde{E}$, we are able to compute $\tilde{E}(\tau \cdot \sigma)$ from $E(\tau)$ and σ. We encrypt quintuples $\sigma = (i_1, ..., i_5)$ by encrypting each element separately. Namely $E(\sigma) = E(i_1), ..., E(i_5)$. To encrypt an element $i \in \{1, 2, ..., 5\}$ we use a quintuple of elements in $Z_m^{(+1)}$ ($m = p \cdot q$ is composed of two large primes), with a quadratic residue in the i-th location and quadratic non-residues in all other locations. Specifically $E_m(i)$ is a probabilistic encryption equaling $(s_1, s_2, ..., s_5)$, where $s_i \in_R Q_m$ and $s_j \in_R N_m$, for all $j \neq i$. For notational convenience we use $E(\cdot)$ instead of $E_m(\cdot)$. For simplicity, we use the shorthand (Q, N, N, N, N) for $E(1)$ ((N, Q, N, N, N) for $E(2)$, etc.), where Q denotes $s \in_R Q_m$ and N denotes $s \in_R (Z_m^{(+1)} - Q_m)$. The advantage of this encryption method is that it allows us to compute an encryption of the Boolean predicate $i = j$ from $E(i)$ and $E(j)$, without yielding any additional information about $i, j \in \{1, 2, 3, 4, 5\}$. Given $E(i)$ and $E(j)$, we first apply coordinate-wise multiplication to the two quintuples, and next apply a random permutation to the result. In case $i = j$, coordinate-wise multiplication yields a quintuple of Quadratic residues. In case $i \neq j$, coordinate-wise multiplication yields a quintuple with Quadratic Non-residues in the i-th and j-th location, and Quadratic residues elsewhere. For example, coordinate-wise multiplication of $E(2) = (N, Q, N, N, N)$ by $E(4) = (N, N, N, Q, N)$ yields (Q, N, Q, N, Q). Applying a random permutation to the result, yields (in case $i \neq j$) a quintuple with two Quadratic Non-residues.

4.1. The Protocol in the Value-Preserving Adversary model.

Preprocessing: B chooses at random two k-bit primes p, q. (k is the security parameter)
B computes $m = p \cdot q$. Next, B chooses $y \in_R N_m$ and publishes the couple m, y.

1) B encrypt (his input) σ, using Quadratic non-residues and residues mod m. (Encryption is as specified above.) B sends $E(\sigma)$ to A.

2) A computes $E(\tau \cdot \sigma)$ by applying (his first input) τ to $E(\sigma)$, as described above.

Let $(i_1, i_2, ..., i_5) = \tau\sigma\tau'$, $(\tau'_1, \tau'_2, ..., \tau'_5) = \tau'$, and $(e_1, e_2, ..., e_5) = E(\tau\sigma)$.
for $j = 1$ to 5 do begin (steps 3.j and 4.j):

3. A computes $\tilde{E}(i_j)$ as follows. First, A picks new probabilistic encryptions $E(1), E(2), ..., E(5)$ (using m and y).
   for $l = 1$ to 5 A computes the coordinate-wise multiplication of e_j and $E(l)$, and randomly permutes the resulting quintuple. Denote the randomly permuted result by $r_{j,l}$.
   A forms five pairs $(r_{j,1}, \tau'_1), (r_{j,2}, \tau'_2), ..., (r_{j,5}, \tau'_5)$, orders the pairs by their rightmost element, and sends the pairs (in this order) to B.

4. B retrieves i_j as follows. Among the five pairs received from A, party B finds a pair with a leftmost element consisting of an "all Quadratic residues" quintuple. B sets σ'_j to be the rightmost element of that pair.

5) B's local output is $\sigma' = (\sigma'_1, \sigma'_2, ..., \sigma'_5)$.

4.2. Correctness of the Protocol

We first show, that the above protocol is correct; namely that the output satisfies the specification conditions.

Claim 4: The permutation σ' locally output by B equals $\tau \cdot \sigma \cdot \tau'$.

Proof: Clearly, in step 2 A correctly computes $E(\tau \cdot \sigma)$ We now show that σ'_j computed by B in step 4.j equals i_j, for every j $(1 \leq j \leq 5)$. The coordinate-wise multiplication of e_j and $E(l)$ equals (Q, Q, Q, Q, Q) if and only if the j-th element of (the quintuple representing the permutation) $\tau\sigma$ equals l. Thus, σ'_j is set to τ'_l, where l is the j-th element of $\tau\sigma$. It follows that σ' equals $(\tau \cdot \sigma) \cdot \tau'$. $\square$

4.3. Maximum Privacy of the Protocol

We now prove that the above protocol has the maximum privacy property. First we use the Intractability Assumption of subsection 2.6 to prove that B preserves privacy with respect to any A^*, and next we prove that A preserves privacy with respect to any B^* The proofs use ideas similar to those used in the proofs of Claims 2 and 3.

Claim 5: Assuming intractability of Quadratic Residuosity (as in subsection 2.4), the above protocol preserves privacy with respect to A.

Proof's Sketch: To prove the claim, we construct a machine M that on input τ, τ', m, y and a restricted polynomial-time computable auxiliary input z, (see (2) at subsection 3.4) outputs the inputs together with the encryption of the identity permutation. As in the proof of Claim 2, ability to distinguish this output from the contents of A's (input and read-only communication) tapes will be converted to a contradiction of the Intractability Assumption of Quadratic Residuosity. $\square$ We now prove preservation of privacy with respect to B. This time we use no intractability assumptions.

Claim 6: The protocol preserves privacy with respect to B.

Motivation to the Proof: What does B learn from the $\tilde{E}(i_j)$'s? Since A uses independently chosen probabilistic encryptions in each step 3.j, we concentrate on what is learned from $\tilde{E}(i_1)$. B has received five pairs, the left element of one of them is (Q, Q, Q, Q, Q) while the left elements of the other pairs are quintuples with three Q's and two N's. It is crucial that the location of the N's in these quintuples is "random" and thus does not leak any information.

Proof' Sketch: We will show that there exist a machine M which on input σ, m, y, z and σ' $(= \tau\sigma\tau')$, outputs a probability distribution which is identical to the distribution on B's input and read-only tapes during interaction with A. M operates as follows

for $j = 1$ to 5 do begin

1. M constructs five pairs $(v_{j,1}, 1), (v_{j,2}, 2), ..., (v_{j,5}, 5)$, where the quintuple $v_{\sigma'_j}$ is (a random) (Q, Q, Q, Q, Q) and all the other quintuples have three (random) quadratic residues in

(independent) randomly selected locations and (random) quadratic non-residues in the remaining locations.

2) M outputs his inputs and all (25) pairs constructed in step 1.

One can easily show that the probability distribution formed by M is identical to the distribution of B when interacting with A. The claim follows. $\square$

4.4. Summing Up

Combining Claims 4, 5 and 6, we get

Theorem 2: The above protocol is a maximum privacy protocol for the permutation switching problem.

ACKNOWLEDGEMENTS

We are grateful to Ilan Newman for listening and commenting on the work, while in progress.

REFERENCES

[Bar] Barrington, D.A., "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1", *Proc. 18th STOC*, 1986, pp. 1-5.

[CGMA] Chor, B., S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *Proc. 26th FOCS*, 1985, pp. 383-395.

[Coh] Cohen, J.D., "Secret Sharing Homomorphisms: Keeping Shares of a Secret", technical report YALEU/DCS/TR-453, Yale University, Dept. of Computer Science, Feb. 1986. Presented in *Crypto86*, 1986.

[DH] Diffie, W., and M.E. Hellman, "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.

[EGL] Even, S., O. Goldreich, and A. Lempel, "A Randomized Protocol for Signing Contracts", *CACM*, Vol. 28, No. 6, 1985, pp. 637-647.

[GMW1] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design", *Proc. 27th FOCS*, 1986.

[GMW2] Goldreich, O., S. Micali, and A. Wigderson, "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority", *Proc. 19th STOC*, 1987.

[GM] Goldwasser, S., and S. Micali, "Probabilistic Encryption", *JCSS*, Vol. 28, No. 2, 1984, pp. 270-299.

[GMR] Goldwasser, S., S. Micali, and C. Rackoff, "Knowledge Complexity of Interactive Proofs", *Proc. 17th STOC*, 1985, pp. 291-304.

[HM] Haber, S., and S. Micali, private communication, 1986.

[Y1] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.

[Y2] Yao, A.C., "How to Generate and Exchange Secrets", *Proc. 27th FOCS*, 1986.