*Programming Techniques and Data Structures*

*Ellis Horowitz Editor*

# A Randomized Protocol for Signing Contracts

SHIMON EVEN, ODED GOLDREICH, and ABRAHAM LEMPEL

*ABSTRACT: Randomized protocols for signing contracts, certified mail, and flipping a coin are presented. The protocols use a 1-out-of-2 oblivious transfer subprotocol which is axiomatically defined.*

*The 1-out-of-2 oblivious transfer allows one party to transfer exactly one secret, out of two recognizable secrets, to his counterpart. The first (second) secret is received with probability one half, while the sender is ignorant of which secret has been received.*

*An implementation of the 1-out-of-2 oblivious transfer, using any public key cryptosystem, is presented.*

## 1. INTRODUCTION

We are rapidly heading into the era of electronic business communication. In the near future, we can expect to see communication networks through which automatic business transactions, such as signing a contract by a pair of computers according to a predetermined protocol, take place. The study of secure protocols for these purposes has recently attracted many researchers (e.g., [3, 5, 8, 10, 19, 23, 33], to list only works which appeared in the recent STOC and FOCS Conferences).

In this article, we consider the problems involved in implementing business transactions, such as signing a contract, in the environment described above. Using a 1-out-of-2 oblivious transfer, we present protocols which solve these problems.

### 1.1 A Specification of a Contract Signing Protocol

Loosely speaking, a contract signing protocol should satisfy the following:

(i) *Viability*: At the end of a proper execution of the protocol, each party has his counterpart's signature to the contract.

(ii) *Concurrency*: If one party X executes the protocol properly, then his counterpart Y cannot obtain X's signature to the contract without yielding his own (i.e., Y's) signature to it.

Using general cryptological assumptions (as the existence of secure public key cryptosystems [6]), we propose a protocol for signing contracts which guarantees the following:

(i) *Viability*: At the end of a proper execution of the protocol, each party can instantly present his counterpart's signature to the contract.

(ii) *Approximate-Concurrency*: If one party X executes the protocol properly, then, with very high probability, at each stage during the execution, X can compute his counterpart's (Y's) signature to the contract using approximately the same amount of work used by Y to compute X's signature to the contract.

We also propose protocols for "certified mail" and "flipping a coin."

### 1.2 Background and Comparison to Previous Work

Even and Yacobi [12] pointed out inherent difficulties which arise in the design of contract signing protocols. Consider protocols in which there exists a transmission such that upon its acceptance a "complete signature" can be computed, whereas before its acceptance not even a "single bit" of the signature can be computed. Even and Yacobi show that such a protocol cannot achieve both viability and concurrency principles.

Trivial protocols for signing contracts are implied by the assumption that reliable third parties take active part in the protocol. However, even under "weaker" assumptions reliable third parties may be useful in the design of contract signing protocols. For example, Rabin [28] pointed out that any *reliable source* of "random noise" is useful in the design of a protocol for signing contracts.[1] Alternatively, a *passive, reliable* "cancellation center" (CC) can be used so that a signed contract is binding only if it has not been cancelled by either party during the cancellation period (specified in the contract).[2]

Even if reliable third parties exist, it is still desirable to have a protocol for signing contracts in which no third party is required. Even [9] proposed a protocol based on Merkle's [24] puzzle concept. His protocol uses any public key cryptosystem deemed secure and relies on the value of the contract's subject. Goldreich [17] proposed a simplified version of Even's protocol. Other protocols, were suggested by Blum [3] and by Blum and Rabin [4]. Their protocols rely on certain number theoretic assumptions (e.g., the infeasibility of factoring large integers).

The main advantages of our protocol over Even's [9] (as well as over Goldreich's [17]) are: (1) Our protocol neither relies on nor makes any reference to the value of the contract's subject; Even's protocol relies, heavily, on this value. (2) Loosely speaking, the quality of our protocol is exponential in its length;[3] the quality of Even's protocol is linear in length.

There are some similarities between our protocol and the protocol suggested, independently, by Blum [3]. Blum's protocol, however, is based on specific number theoretical assumptions (one of which has recently been shown false [22]); whereas our assumptions are of universal cryptological nature (e.g., the existence of a secure PKCS).

Our contract signing protocol uses a 1-out-of-2 oblivious transfer ($OT_2^1$) as a subroutine. The notion of oblivious transfer (OT) was introduced by Rabin [27], in a number theoretic context. Rabin also presented an implementation of OT based on the factoring problem.[4] In this article, we present what we believe to be a more natural definition of OT. We present an implementation of OT using any public key cryptosystem. An axiomatic definition of $OT_2^1$ as well as its implementation are also presented.

### 1.3 The Organization of this Article

Section 2 contains the assumptions and some of the notations used throughout this article. These assumptions are of pure cryptological nature. In Section 3 we give formal definitions of OT and 1-out-of-2 OT. Implementations of both transfers are given in Section 8.

In Section 4 we present a subprotocol which will serve as the core of our contract signing protocol. This subprotocol allows two parties to exchange two sets of pairs of secrets such that each party will get at least one pair of his counterpart's secrets. In Section 5 we analyze this subprotocol and show that the probability of successfully cheating is exponentially small (in the number of pairs).

In Section 6 we present our contract signing protocol. In Section 7 we present a "certified mail" protocol and a "coin flipping" protocol, both using the same subprotocol of Section 4.

### 1.4 On the Notions of Certified Mail and Flipping a Coin

The classical (everyday) notion of sending certified mail means a procedure through which the receiver gets the mail if and only if the sender gets a receipt (which certifies the fact that the receiver got mail from the sender at a specific time). This procedure, although used for centuries, is weak since the receipt does not certify the contents of the received mail. We follow Blum's definition of *sending certified electronic mail* [1]. By this definition, the receiver gets the mail if and only if the sender gets a receipt which certifies *the contents of the mail.*

By "flipping a coin" we mean not only that the outcome (of the coin-flip) will be random and unforgeable

[1] Rabin assumes that any user can get the value of the noise transmitted in the past; but no user can predict the value to be transmitted in the future.

[2] To sign a contract, the first party sends his signature to the second, who responds with his signature. If the second party does not respond within a reasonable time, then the first party sends a cancellation message to the CC.

[3] By the length of the protocol we mean the number of transmissions exchanged in it.

[4] Rabin's implementation, although based on the factoring problem, was not proven to be "equivalent" to it. This was observed by Fischer, Micali, and Rackoff [13] who recently modified Rabin's implementation and proved that their implementation is "equivalent" to factoring. That is, one can cheat when executing their implementations if and only if one can factor large integers. For further details, consult [20].

but also that one party knows it if and only if his counterpart knows it. Concurrent knowledge of the outcome was not required (and was not achieved) in [2].

## 2. ASSUMPTIONS

Our assumptions are partitioned into *general assumptions*, relied on throughout this article, and *OT's implementation assumption*, used (only) in our implementation of the oblivious transfer (i.e., in Section 8).

### 2.1 General Assumptions

1. *Equal Resources Assumption*: The computational capabilities of both parties are (approximately) the same.

Note that we do not assume here "equal knowledge of algorithms."

2. *Secure Public-Key Signature Scheme Assumption*: There exists a secure public-key signature scheme (PKSS).

Several PKSS's which meet various security criteria and are based on various intractability assumptions have been proposed [21, 26, 30]. The PKSS presented in [21] is the most robust and is based on the weakest intractability assumption. However, we believe that using the RSA as a PKSS will be secure enough for our purposes.

3. *Uniformly-Secure Conventional Cryptosystem Assumption*: There exists a "uniformly-secure" conventional cryptosystem (see below for a definition). Let $F$ denote a conventional cryptosystem (e.g., the DES [25]). By $F_K(pln)$ we denote the encryption of the plaintext $pln$ using $F$ with key $K$. By $F_K^{-1}(cip)$ we denote the decryption of the ciphertext $cip$ using $F$ with key $K$. We say that $F$ is uniformly-secure (if $F$ is secure and) if the expected time of computing the key $K$, given a plaintext-ciphertext pair $(M, F_K(M))$ and the first $i$ bits of $K$, is invariant of the values of the first $i$ bits of the key.

Let us be more precise. $F$ is *uniformly-secure* if:

(i) It is infeasible to compute $K$ when given only the plaintext-ciphertext pair $(M, F_K(M))$.

For a given input consisting of a pair $(M, C)$ and an $i$-bit string $s$, $K$ is called a *solution* if $F_K(M) = C$ and the first $i$ bits of $K$ agree with $s$. Let $t_A(s)$ denote the expected time in which Algorithm A finds a solution, for a given $(M, C)$ and $s$. (The average running time is taken over all possible inputs which agree with $s$.)

(ii) There exists an ("optimal") Algorithm A for which $t_A(s)$ depends only on the number of unknown bits in $K$; that is, $T_A(k - i) = t_A(s)$ for each $i$-bit string $s$, where $k$ denotes the length of $K$. Furthermore, this algorithm is known to all parties.

(iii) There exists no Algorithm A′ such that for all sufficiently large $k$ and for a nonnegligible fraction of the keys of length $k$, Algorithm A′ finds a solution in expected time $\frac{1}{2}T_A(k - i)$. (Again, the average running time is taken over all possible inputs which agree with $s$.)

(iv) Algorithm A, on input $(M, C)$ and an $i$-bit string $s$, for which there is no solution, reaches this conclusion in at most $2T_A(k - i)$ time.

Note that (ii) above implies equal knowledge of algorithms in a very restricted sense: public knowledge of an optimal "breaking algorithm" for $F$. (For simplicity, the reader may further assume that this optimal algorithm consists of the trivial exhaustive search over the key space. However, this simplifying assumption is not essential.)

We believe that the DES is "uniformly secure" in the sense that it is possible to find the unknown bits of the key $K$, when given some of the bits of $K$ and the plaintext-ciphertext pair $(M, DES_K(M))$, only by an exhaustive search over the subspace of all keys which agree with the given bits of $K$.

4. *$OT_2^1$ Existence Assumption*: There exists a protocol (hereafter referred to as $OT_2^1$) which satisfies the axioms given in Section 3.2.

This assumption can be substituted by Assumption 4′ (below) which, in turn, implies the existence of $OT_2^1$.

### 2.2 OT's Implementation Assumption

The following assumption (concerning public-key cryptosystems) is optional. It implies Assumption 4 and is relied on only in our implementation of $OT_2^1$ (see Section 8).

A public-key cryptosystem (PKCS) [6] consists of three algorithms: a key generation algorithm G, an encryption algorithm E and a decryption algorithm D. On input $x$, G generates a pair $(e_x, d_x)$ of encryption decryption keys. Let $E_x(M)[D_x(M)]$ denote the encryption [decryption] of message $M$ using the encryption key $e_x$ [the decryption key $d_x$].

Let $\mathscr{X}$ denote the set of all possible inputs to the key-generating algorithm and $\mathscr{M}_x$ denote the message space for the PKCS instance generated by input $x$. We further assume that for every $x \in \mathscr{X}$ and every $m \in \mathscr{M}_x$, $E_x$ and $D_x$ are defined and are inverse operators; that is,

$$E_x(D_x(m)) = D_x(E_x(m)) = m.$$

In other words, $\{E_x, D_x: x \in \mathscr{X}\}$ is the set of the PKCS *operators* and $\{D_xE_x \equiv \lambda, E_xD_x \equiv \lambda: x \in \mathscr{X}\}$ is the set of the operators' *cancellation rules*. (Compare [7].)

4′. *PKCS's Freeness Assumption*: Loosely speaking, we would like to assume that an adversary cannot use, in his computations, any "relations" between the PKCS operators which are not implied by the operators' cancellation rules. Giving a precise formulation of this "freeness assumption" seems to be difficult. We avoid it by suggesting a specific freeness assumption, with respect to the PKCS in use (for details see Section 8).

Recently, Rackoff and Luby [29] used the random-function construction [18] to show that private-key cryptosystems which satisfy the above "freeness as-

sumption" do exist (if any one-way permutations exist). This provides some additional support to our belief in the validity of Assumption 4'.

## 3. OBLIVIOUS TRANSFER AND 1-OUT-OF-2 OBLIVIOUS TRANSFER

In this section we present axiomatic definitions of oblivious transfer (OT) and 1-out-of-2 oblivious transfer. Implementations of these transfers are given in Section 8.

The notion of a "recognizable secret message" plays an important role in our definition of OT. A message is said to be a recognizable secret if, although the receiver cannot compute it, he can authenticate it once he receives it. A formal definition of a recognizable secret message is proposed in the Appendix. Following are three common examples of recognizable secret messages:

(i) A signature of a user to some known message is a recognizable secret message for everybody else.

(ii) The key $K$, by which the plaintext $M$ is transformed using cryptosystem $F$ into ciphertext $F_K(M)$.

(iii) The factorization of a composite number, which has only large prime factors. (E.g., the factorization of an RSA modulus [30].)

The notion of a recognizable secret message is evidently relevant to the study of cryptographic protocols, in which the sender is reluctant to send the message while the receiver wishes to get it. In such protocols, it makes *no sense to consider the transfer of messages that are either not secret (to the receiver) or not recognizable (by the receiver).*[5] We believe that in any reasonable application of OT [1-out-of-2 OT] the receiver is reluctant to send and the sender wishes to receive the message. Thus, we consider only the transfer of messages which are recognizable secrets (for the receiver).

### 3.1 Oblivious Transfer

An *oblivious transfer* (OT) of a recognizable secret message $M$ is a protocol by which a *sender* $S$ transfers to a *receiver* $R$ the message $M$ so that $R$ gets $M$ with probability one half while for $S$ the a-posteriori probability that $R$ got $M$ remains one half. Note that OT is defined for any kind of recognizable secret messages.

The OT defined above has three parameters: $S$, $R$, $M$; it will be hereafter denoted by OT($S$, $R$, $M$). To sum up, the protocol OT($S$, $R$, $M$) has to satisfy the following three axioms:

(i) If $S$ executes OT($S$, $R$, $M$) properly, then $R$ can read $M$ with (a-priori) probability (exactly) one half. Furthermore, in case $R$ does not read $M$, he gains (by the execution of OT($S$, $R$, $M$)) no "helpful partial information" about $M$ in the following sense: Assume that, after the execution of OT($S$, $R$, $M$), $R$ is given the value of a predetermined function of $M$ (e.g., the first five bits of $M$). Then computing $M$ is not easier than in the case $R$ is given this value without first executing OT($S$, $R$, $M$).

(ii) For $S$, the a-posteriori (i.e., after the execution of OT($S$, $R$, $M$)) probability that $R$ can read $M$ remains one half, provided $S$ and $R$ have executed OT($S$, $R$, $M$) properly.

(iii) If $S$ tries (to deviate from the protocol in order) to decrease the probability that $R$ will get $M$, then $S$ can detect this attempt with probability at least one half. The same (detection with probability one half) holds in case $S$ tries to deviate from the protocol in order to increase his a-posteriori probability of guessing whether $R$ read $M$ (without increasing the probability that $R$ reads $M$).

We believe that axiom (iii) cannot be strengthened; namely

*Conjecture*: If axiom (iii) is changed to require detection with probability greater than one half of attempts to cheat (i.e., to prevent $R$ from getting $M$), then there exists no implementation which satisfies all the axioms.

Note that Rabin's oblivious transfer [27] is not a counterexample to our conjecture. True, in Rabin's OT, $S$ cannot cheat without being detected. But the message transferred by Rabin's OT is restricted to be a secret inherent to this transfer[6] rather than an arbitrary recognizable secret message. Indeed, one can easily modify Rabin's OT so that an arbitrary recognizable secret message can be transferred by it.[7] However, in the modified protocol, attempted cheating is detected with probability one half.

### 3.2 The 1-out-of-2 Oblivious Transfer

A *1-out-of-2 oblivious transfer* ($OT_2^1$) is a protocol by which a *sender* $S$ transfers ignorantly to a *receiver* $R$ one message out of two recognizable secret messages. More precisely, an $OT_2^1(S, R, M_1, M_2)$ is a protocol that satisfies the following three axioms:

(i) If $S$ executes $OT_2^1(S, R, M_1, M_2)$ properly, then $R$ can read exactly one message: either $M_1$ or $M_2$; the probability of each to be read is one half. Furthermore, in case $R$ does not read $M_i$, he gains (by the execution of $OT_2^1$) no "helpful partial information" about $M_i$, $i \in \{1, 2\}$ (see Section 3.1).

(ii) For $S$, the a-posteriori probability that $R$ got $M_1$ remains one half, provided $S$ and $R$ have executed $OT_2^1$ properly.

(iii) If $S$ tries to (deviate from the protocol in order to) increase $S$'s a-posteriori probability of guessing which message was read by $R$, then $R$ can detect this attempt with probability at least one half.

Note that OT can be implemented using any implementation of $OT_2^1$ (by letting $M_2$ be a message known to both parties). That is,

$$OT(S, R, M) = OT_2^1(S, R, M, K),$$

where $K$ is a-priori known to $R$.

However, it is not clear whether $OT_2^1$ can be implemented using any OT (e.g., using Rabin's OT [27]).

[5] If the message is not secret (to the receiver), then the receiver either knows it or can feasibly compute it before the transfer takes place. If the message is not recognizable (by the receiver), then "getting it" means nothing. In both cases, participation in the transfer protocol does not provide the "receiver" with something he cannot get by himself.

[6] Rabin's OT transfers the factorization used by the OT itself.

[7] The obvious way, to modify Rabin's OT, is to encrypt the recognizable secret using an instance of the RSA which has the same modulus as the modulus used by the OT.

Other generalizations of the oblivious transfer (e.g., biased oblivious transfer and one-out-of-many oblivious transfer) were suggested by Goldreich [15]. These generalizations were implemented (similarly to the implementation given in Section 8) and were used as protocol design tools (see [15]).

## 4. THE PARTIAL SECRETS EXCHANGE SUBPROTOCOL (PSE)

The following subprotocol will be used in the transaction protocols presented in Sections 6 and 7. The parties to the subprotocol will be called $A$ and $B$. It is assumed that $A$ holds $2n$ secrets, denoted $a_1, a_2, \ldots, a_{2n}$, all recognizable by $B$. Similarly, $B$ holds $2n$ secrets, $b_1, b_2, \ldots, b_{2n}$, recognizable by $A$. The secrets are assumed to be binary strings of length $l$.

The secrets of each party are partitioned into pairs: $A$'s pairs are $(a_1, a_{n+1}), (a_2, a_{n+2}), \ldots, (a_n, a_{2n})$; and $B$'s pairs are $(b_1, b_{n+1}), (b_2, b_{n+2}), \ldots, (b_n, b_{2n})$.

We say that $A$ [$B$] *effectively-knows one of B's* [$A$'s] *pairs* if there exists an $i$, $1 \leq i \leq n$, such that $A$ [$B$] can efficiently compute both $b_i$ and $b_{n+i}$ [$a_i$ and $a_{n+i}$].

The purpose of the subprotocol is to exchange effective-knowledge of any one pair of secrets. This subprotocol will hereafter be referred to as the *Partial Secret Exchange* (PSE) subprotocol.

### The Partial Secret Exchange Subprotocol

```
protocol
  PSE(A, B, {(a_i, a_{n+i})}_{i=1}^{n}, {(b_i, b_{n+i})}_{i=1}^{n})
(step 1)
for i = 1 to n do begin
    OT_2^1(A, B, a_i, a_{n+i})
        (A sends a_i and a_{n+i} to B via OT_2^1)
    OT_2^1(B, A, b_i, b_{n+i})
        (B sends b_i and b_{n+i} to A via OT_2^1)
end;
(Comment: At this stage each party X has
exactly one element of each pair of his
counterparts' secrets; while his coun-
terpart is ignorant of which elements X
knows.)
(step 2)
for j = 1 to l do begin
(l is the length of each of the secrets)
    A transmits the jth bit of each a_i
    to B (1 ≤ i ≤ 2n)
    B transmits the jth bit of each b_i to
    A (1 ≤ i ≤ 2n)
end;
```

To avoid being cheated, each party $X$ should take the following precautions:

(i) During step 1, while playing the role of the receiver in $OT_2^1$, ($X$ should) use the "cheat-detection mechanism" of $OT_2^1$. (The existence of this "mechanism" is guaranteed by axiom (iii) of $OT_2^1$.)

(ii) While executing step 2 ($X$ should) check whether the bits revealed (to him) during the alternating substeps match the bits of the secrets which have been disclosed (to him) in step 1.

A party should stop further execution of the protocol as soon as he detects an attempt to cheat. This is sufficient to protect oneself against cheating.

### Remarks

1. Note that if both parties execute PSE properly, then each will have all his counterpart's secrets. In the next section we show that if a party $X$ executes PSE properly then, with very high probability, he is guaranteed to effectively know at least one of his counterpart's pairs in case his counterpart effectively knows one of $X$'s pairs.

2. The interleaving in step 1 is not essential. That is, one can first execute all the instances of $OT_2^1$ in which $A$ plays the role of the sender and only then execute the instances in which $A$ is the receiver.

3. In an earlier version of this article [11] OT was used in step 1 of the protocol instead of $OT_2^1$; that is, only one secret was transferred in every loop (and it was received with probability one half). This made the analysis of the subprotocol more complicated. Also, the use of the previous version of the subprotocol [11] to send certified mail had required a threshold scheme (e.g., Shamir's scheme [31]).[8]

4. Consider the case in which a proper execution of PSE is terminated at a "late stage" of step 2 and assume each party can compute one of his counterpart's pairs. If this happens after $A$ has transmitted the $j$th bit of each $a_i$, but before $B$ has answered (with the $j$th bit of each $b_i$) then: the expected time $A$ has to invest (in order) to get one of $B$'s pairs is twice as much as the expected time $B$ needs (in order to get one of $A$'s pairs). If this advantage is considered to be excessive, then one can use the simple "exchange of half a bit" suggested by Tedrick [32].

5. To make it possible for an honest party to not only protect himself against cheating but also prove that his counterpart has cheated, signatures should be provided to all transmissions of the subprotocol. (This includes the transmissions of $OT_2^1$).

## 5. ANALYSIS OF PSE

As mentioned in the previous section, if both parties follow PSE properly to its conclusion, then each will have all his counterparts' secrets. Furthermore, in such a case during the execution of step 2 each party can compute one of his counterpart's pairs spending at most twice as much expected time as needed by his counterpart to compute one of his pairs. In this section we consider the case in which one party $X$ follows PSE properly while the other deviates from it. We will show that in such a case $X$ is guaranteed that if $Y$ tries to effectively know one of $X$'s pairs then, with very high probability, $X$ will be able to compute a pair of $Y$'s in about the same effort.

Let us denote $X$'s [$Y$'s] $i$th pair, by $(x_i, x_{n+i})$ [$(y_i, y_{n+i})$]. In our proofs we will rely on the existence of $OT_2^1$ (Assumption 4).

[8] For further details, on the previous version of the PSE (which was called the Majority Exchange) and on the way to use it (in order) to send certified mail, consult [16].

LEMMA 1

*If X executes PSE properly, then after step 1 is concluded the following hold:*

*(i) Y knows a single element out of each of X's pairs.*

*(ii) X knows (at least) one element out of each of Y's pairs.*

PROOF

Part (i) follows from axiom (i) of $OT_2^1$.

Part (ii) follows from the fact that X would have terminated the execution of PSE, had he not received one element out of each of Y's pairs, during step 1. □

Thus, before step 2 is executed, Y has no effective knowledge of any of X's pairs; provided X follows PSE properly. We remind the reader that by saying that a party effectively knows a value we mean that he can compute it efficiently. We say that a party *T-knows* a value if he can compute it in "PSE-based expected time" T, to be defined below.

Here, and throughout this section, we consider computations (of secrets) based on input (information) which is partially a-priori known and partially obtained throughout the execution of PSE. These computations will take place at an arbitrary point during the execution of PSE. By the *expected time of a computation given an instance of PSE* (*PSE-based expected time*) we mean that the average running time is taken over all inputs which agree with the values disclosed in the substeps (of step 2 of PSE) which have already been executed. We provide motivation to this defintion in Remark 2 following the proof of the Theorem.

In the rest of this section, we will assume that the secrets are *uniformly hard* in the following sense: (Compare Assumption 3.)

(i) There exists an (optimal) Algorithm A which computes the secret, when given $i$ out its $l$ bits, in expected time (exactly) $T_A(l - i)$. (Here and in (ii) and (iii) below, the average running time is taken over all possible secrets which agree with the given $i$ bits.) Furthermore, this algorithm is known to all parties.

(ii) There exists no Algorithm A′ such that: for all sufficiently large $l$ and for a nonnegligible fraction of all possible secrets of length, $l$, A′ computes the secret, when given $i$ out of its $l$ bits, in expected time $\frac{1}{2}T_A(l - i)$.

(iii) The optimal Algorithm A, when given an $i$-bit string $s$, which matches no secret, reaches this conclusion in at most $2T_A(k - i)$ time.

Loosely speaking, this means that the expected time of computing the secret, given the values of $i$ of its bits, is invariant of the values of *these bits.*

LEMMA 2

*Suppose that X executes PSE properly and that step 1 has been concluded. If Y deviates from PSE, in order to reach a situation in which Y T-knows one of X's pairs but X does not 4T-know Y's ith pair, then X can detect this (cheating attempt) with probability at least one half.*

PROOF

Consider the situation in which Y T-knows one of X's pairs but X does not 4T-know Y's $i$th pair. Without loss of generality, let $i = 1$. First, we show that the probability that Y can speed-up his computation (with respect to the optimal algorithm mentioned in (i) above) is negligible:

By axiom (i) of $OT_2^1$, the PSE-based expected time of computing a secret, which was not disclosed in step 1 of PSE, depends only on the bits (of the secret) which were disclosed during the alternating substeps of step 2.

By the uniform hardness of the secrets, if Y can compute one of X's pairs in PSE-based expected time $T$, then Y can compute it in PSE-based expected time at most $2 \cdot T$ using the optimal algorithm.

Finally, note that using the optimal algorithm the PSE-based expected time of computing a secret does not depend on the values (of the secret) disclosed during step 2. (Thus, analyzing the performance of the optimal algorithm, it is sufficient to refer to the number of the secret's unknown bits.)

Thus, we may assume (without loss of generality) that Y can compute one of X's pairs in expected time $2 \cdot T$ using the optimal algorithm which is publically known but X cannot compute Y's first pair in expected time $2 \cdot 2T$ using the same algorithm. Such a situation could not occur if Y had transmitted to X, during the alternating substeps of step 2, the true values for the appropriate bits of both $y_1$ and $y_{n+1}$ (recall Assumption 1: the parties have equal computing power). Therefore, in order to reach such a situation Y must commit *action 2* and may commit *action 1*, where:

*action 1* is to cheat[9] in $OT_2^1(Y, X, y_1, y_{n+1})$ which takes place at step 1.

*action 2* is to transmit, during the alternating substeps of step 2, either false values for the bits of $y_1$ or false values for the bits of $y_{n+1}$.

By axiom (iii) of $OT_2^1$, X can detect action 1 with probability at least one half.

If action 1 was not committed, then by axiom (i) of $OT_2^1$, X got either $y_1$ or $y_{n+1}$ during step 1. Also note that, by axiom (ii) of $OT_2^1$, Y cannot guess, with probability greater than one half, which $y$ has been received by X. Thus, if Y attempts to commit action 2 then X will detect it with probability one half. (Since if Y tries to give wrong values for the bits of the $y$ that X knows then X trivially detects an attempt to cheat.) □

THEOREM

*If X executes PSE properly and step 1 has been concluded, then*

*(i) If Y deviates from PSE, in order to reach a situation in which Y T-knows one of X's pairs but X does not 4T-know any of Y's pairs, then X can detect this (cheating attempt) with probability at least* $1 - 2^{-n}$. *Furthermore, if Y deviates from PSE, in order to reach a situation in which Y T-knows*

[9] Here, cheating in the $OT_2^1$ means, as in Section 3, trying to increase the a-posteriori probability of guessing which message was read by the receiver.

*one of X's pairs but for at least half of the $i \in \{1, 2, \ldots, n\}$ X does not 4T-know Y's ith pair, then X can detect this with probability at least $1 - 2^{-n/2}$.*

*(ii) If Y does not try to reach the latter situation described in part (i) then if Y T-knows one of X's pairs then X can compute one of Y's pairs, in PSE-based expected time 16T. Furthermore, with probability at most $(\frac{1}{2})^{-j}$, X will spend more than 8jT expected time in computing one of Y's pairs.*

PROOF

Part (i) follows from Lemma 2.

Part (ii): We say that Y's $i$th pair is *good* if X can compute it in expected time $4T$, using the optimal algorithm; otherwise we say that this pair is *bad*. Note that more than half of Y's pairs are good; however X does not know which ones are good. Nevertheless, X can choose one of Y's pairs at random and try to compute it. This computation takes at most $8T$ time, after which X either has the pair or does not have it (recall the uniform-hardness assumption). Recall that with probability at least one half the pair is good and thus X has it. If X does not have that pair, he will choose another pair. Again with probability at least one half, X chose a good pair and will have it after conducting the computation. This goes on until X chooses a good pair and has it (after computing it). Note that the probability that X will spend more than $8jT$ time, while following the above procedure, is less than $(\frac{1}{2})^{-j}$. The reader can verify that following this procedure X will have one of Y's pairs in expected time $16T$.[10] □

**Remarks**

1. Note that by Lemma 1, X is "protected" during his proper execution of step 1; while by the Theorem, X is "protected" during his proper execution of step 2. This implies that PSE satisfies the approximate-concurrency principle with respect to computation of one of the counterpart's pairs. By Remark 1 in Section 4, PSE also satisfies the viability principle.

2. Loosely speaking, the theorem implies that the PSE-based expected time of computing one of the counterpart's pairs is about the same for both parties. The fact that it is the PSE-based expected time (which is the same for both parties) and not the expected time averaged over all inputs, is crucial. This means that approximate concurrency is achieved in every PSE execution and not just when averaging over all PSE executions.

## 6. THE CONTRACT SIGNING PROTOCOL

Let $C$ be a contract and $A$ and $B$ be the parties to it. The contract has been negotiated and informally agreed upon. The role of the following protocol is to allow $A$ and $B$ to exchange formal signatures to the contract.

The essence of the proposed protocol is that each party X will randomly generate a set of pairs of secrets and will declare that he (X) is committed to the contract if his counterpart knows one of his (X's) pairs. These sets of pairs will be exchanged by PSE, presented in Section 4.

We remind the reader that $F$ denotes the (uniformly-secure) conventional cryptosystem in use (see Section 2). From now on, the key $K$ is said to be a *solution of the $S_o$-puzzle $P_o$*, if $P_o = F_K(S_o)$.

The message $S$, used in the protocol, is an arbitrary standard message.

**The Contract Signing Protocol**

(step 1)
$A$ generates, randomly, $2n$ keys $(a_1, a_2, \ldots, a_{2n})$ to $F$;
$A$ computes $C_i^A = F_{a_i}(S)$, $1 \le i \le 2n$;
$A$ declares that
  [Beginning of A's Declaration]
  [Denotation:] The symbols $K_1^A, K_2^A, \ldots, K_{2n}^A$ denote solutions of the corresponding $S$-puzzles: $C_1^A, C_2^A, \ldots, C_{2n}^A$.
  [Statement:] I am committed to the contract $C$ if $B$ can present both $K_i^A$ and $K_{n+i}^A$, for some $1 \le i \le n$. (I.e., both solutions of the $i$th and $(n+i)$th puzzles.)
  [End of A's Declaration]
$A$ signs this declaration and transmits it to $B$;
$B$ acts symmetrically, generating the keys $b_1, b_2, \ldots, b_{2n}$ and denoting by $K_i^B$ the solution of the $S$-puzzle $F_{b_i}(S)$;
(Comment: At this stage each party $X$ has a declaration, signed by $X$'s counterpart $(Y)$, which specifies (i.e., determines) what will be considered as $Y$'s signature to the contract $C$.
However, the computation of this signature is infeasible.)
(step 2)
$\mathrm{PSE}(A, B, \{(a_i, a_{n+i})\}_{i=1}^n, \{(b_i, b_{n+i})\}_{i=1}^n)$;
(I.e., $A$ and $B$ apply PSE to exchange the sets $\{(a_i, a_{n+i})\}_{i=1}^n$ and $\{(b_i, b_{n+i})\}_{i=1}^n$).

**Remarks**

1. After step 1 the $a_i$'s are recognizable secret messages for $B$. Similarly the $b_i$'s are recognizable secret messages for $A$. We chose to make the $a_i$'s recognizable, by using them as solutions of the $S$-puzzles. Note that the $a_i$'s remain secret due to the security of $F$ (see Assumption 3). Any other method, to make the $a_i$'s recognizable (yet still secret and uniformly hard), will do as well. We believe that the DES [25] can be used as $F$ in our protocol.

[10] Let $T'$ denote the worst-case time of conducting the above defined computation. Then

$$T' = 8T\left(\left(\frac{1}{2}\right)\frac{n-k}{n} + \left(1+\frac{1}{2}\right)\frac{k}{n}\frac{n-k}{n-1} + \left(2+\frac{1}{2}\right)\frac{k}{n}\frac{k-1}{n-1}\frac{n-k}{n-2} + \cdots + \left(k+\frac{1}{2}\right)\frac{k}{n}\frac{k-1}{n-1}\cdots\frac{1}{n-(k-1)}\frac{n-k}{n-k}\right),$$

where $k \le (n/2)$ is the number of bad pairs. Note that $T' < 8T((\frac{1}{2}) + 2(\frac{1}{2})^2 + 3(\frac{1}{2})^3 + \cdots)$. Thus, $T' < 16T$.

2. Recall that we have assumed the existence of a secure PKSS (Assumption 2). Note that the statements, *A* and *B* have signed and sent each other reduce the "problem" of having a signature (of the counterpart) to the contract *C* into the "problem" of knowing one of the counterpart's pairs. Applying PSE "solves" the latter "problems," concurrently, for both parties. Thus, if *X* follows the above protocol properly and his counterpart *Y* can compute *X*'s signature to *C* then with very high probability *X* can compute *Y*'s signature to *C*, spending about the same amount of work. (This, as well as other properties of the above protocol, is induced by PSE's properties.)

3. An important feature of the above protocol is that with very high probability $(1 - 2^{-n})$ at any moment, in which it is *feasible* to compute a signature to the contract, both parties can do so by spending approximately the same amount of time. This is because computing the signature becomes feasible only during step 2 of the PSE. At that point each party knows with very high probability that he has the information which allows this computation. The above feature is absent from Even's protocol [9] as well as from Goldreich's protocol [17]. In their protocols, the information which allows a feasible computation of *A*'s signature to the contract reaches *B* before *A* gets anything from *B*. (The point in their protocols is that this computation, although feasible, is not beneficial.)

## 7. CERTIFIED MAIL AND FLIPPING A COIN

Using ideas similar to those of the previous section, we will present protocols for 'certified mail' and for 'flipping a coin.'

### 7.1 A Protocol for Certified Mail

Let *M* denote a message *A* wants to send to *B*, via certified mail. We remind the reader that *A* would like a receipt which certifies that the mail has been received by *B*. *B* does not know *M* and is to get it if and only if *A* gets *B*'s acknowledgment to the fact that he (*B*) has got *M*.

The essence of the proposed protocol is that *A* first transmits an encryption of the mail to *B* and *B* acknowledges having received this encryption. Let *K* denote the key used in the encryption of *M*. The certified mail protocol is thus reduced to an exchange of the key *K*, for a receipt which specifies *K*. We implement the key-receipt exchange, using ideas similar to those of the previous section.

Each party will randomly generate a set of pairs (of recognizable secrets). Having any one of *A*'s pairs will yield the key *K*. Having any one of *B*'s pairs will be part of the receipt; the other part of the receipt will be a proof that *A* has chosen his pairs so that they satisfy the above (i.e., that each pair yields *K*). These sets of pairs will be exchanged by PSE. The validity of the certified mail protocol follows from the properties of PSE.

We remind the reader that *F* denotes the conventional cryptosystem in use (see Section 2). We further assume here that it is infeasible to find *M*, $K_1$, and $K_2$ such that $F_{K_1}(M) = F_{K_2}(M)$. As in Section 6, the key *K* is said to be a *solution of the $S_o$-puzzle $P_o$* if $P_o = F_K(S_o)$. The message *S*, used in the protocol, is an arbitrary standard message.

**The Certified Mail Protocol**

(step 1A)
*A* generates, randomly, $n + 1$ keys $(a_0, a_1, a_2, \ldots, a_n)$ to *F*;
*A* computes $a_{n+i} = a_0 \oplus a_i$, for $1 \le i \le n$, where $\oplus$ denotes the bit-by-bit addition modulo 2;
*A* computes $C = F_{a_0}(M)$ and $C_i^A = F_{a_i}(S)$, $1 \le i \le 2n$;
*A* transmits *C* and $C_1^A, C_2^A, \ldots, C_{2n}^A$ to *B*;
(Comment: At this stage *B* has an encryption of the mail.)
(step 1B)
*B* generates, randomly, $2n$ keys $(b_1, b_2, \ldots, b_{2n})$ to *F*;
*B* computes $C_i^B = F_{b_i}(S)$, $1 \le i \le 2n$;
*B* declares that
  [Beginning of *B*'s Declaration]
  [Denotation:]
  The symbols $K^A$, $K_2^A, \ldots, K_{2n}^A$;
  $K_1^B, K_2^B, \ldots, K_{2n}^B$
  denote the solutions of the corresponding *S*-puzzles:
  $C_1^A, C_2^A, \ldots, C_{2n}^A$; $C_1^B, C_2^B, \ldots, C_{2n}^B$.
  The symbol $K_o^A$ denotes a key to *F* ($K_o^A$ must satisfy (2) below).
  [Statement:] I acknowledge having received the mail, which results from decrypting *C* by *F* using the key $K_o^A$, if *A* can present the following (i.e., both (1) and (2)):
  (1) Both $K_i^B$ and $K_{n+i}^B$, for some $1 \le i \le n$.
  (2) $K_j^A$, for all $1 \le j \le 2n$, so that for every $i$, $1 \le i \le n$, $K_o^A = K_i^A \oplus K_{n+i}^A$
  [End of *B*'s Declaration]
*B* signs this declaration and transmits it to *A*;
(Comment: At this point *A* has a declaration, signed by *B*, which specifies (i.e., determines) what will be considered as *B*'s acknowledgment to having received the mail.
However, the computation of this ''receipt'' is infeasible.)
(step 2)
PSE(*A*, *B*, $\{(a_i, a_{n+i})\}_{i=1}^n$, $\{(b_i, b_{n+i})\}_{i=1}^n$);
(I.e., *A* and *B apply PSE to exchange the sets* $\{(a_i, a_{n+i})\}_{i=1}^n$, and $\{(b_i, b_{n+i})\}_{i=1}^n$).

**Remarks**

1. Note that knowing one element of each of *A*'s pairs does not allow the determination of $a_0$; not even

from an information theoretic point of view. On the other hand, knowing both elements of one of $A$'s pairs allows a fast computation of $a_o$.

2. As noted by Goldreich [14], any protocol for sending certified mail can be used for mail disclosure (i.e., a transfer of information such that its use by the receiver is limited to specific terms which were agreed upon, before the receiver has received the information). It was also noted that contracts can be signed by the use of certified mail.

3. The idea of reducing the certified mail protocol to a "key-receipt exchange" was used by Goldreich [14] in his simpler protocol for certified mail. His protocol (by which $A$ sends to $B$ the certified mail $M$) proceeds as follows:

notation: $S_B(M)$ denotes the signature of $B$ to the message $M$.)
(step 1A) $A$ chooses, randomly, a key $K$ to $F$;
$A$ transmits $F_K(M)$ and $F_K(S)$ to $B$;
(Comment: At this stage $B$ has the encryption of the mail as well as an $S$-puzzle, the solution of which is the key used for encrypting the mail.)
(step 1B) $B$ transmits $S_B$('*from A*', $F_K(M)$, $F_K(S)$) to $A$;
(Comment: $B$ acknowledges having received the above.)
(step 2) [The key-receipt exchange]
***for*** $i = 1$ to $l$ ***do begin***; [$l$ denotes the length of $K$.]
$A$ transmits, $k_i$, the $i$th bit of $K$ to $B$;
$B$ transmits $S_B(F_K(M), i, k_i)$ to $A$;
***end***;
(Comment: $B$ acknowledges each bit of the key he gets.)

Note, that the sender can prove a tight upper bound on the amount of work the receiver has to invest in order to read the mail. However, the above protocol is unsymmetric in the following sense. If the execution is prematurely terminated, then the sender can instantaneously present this proof, while the receiver must invest time to read the mail. This unsymmetry is not present in our protocol. In case our protocol is prematurely terminated, both parties need to invest work to achieve their goals.

### 7.2 A Protocol for Flipping a Coin

Let $A$ and $B$ be two parties who wish to conduct a coin flip (through the network). We remind the reader that each party X would like to be guaranteed that his counterpart $Y$ can neither change the outcome of the coin-flip nor distinguish it from a truly random coin-flip. Also, $X$ wants to know the outcome (of the coin-flip) at least as soon as $Y$ knows it.

The essence of the proposed protocol is that the outcome (of the coin-flip) will be the parity of 'a certified mail $A$ sends to $B$ and a certified mail $B$ sends to $A$.' Note that it is important that both parties get the certified mail sent to them concurrently. To this end, two instances of the certified mail protocol, of the previous subsection, are interleaved.

Let $s = (s_1, s_2, \ldots, s_m)$ be a binary string of length $m$. We denote by Par($s$) the number of ones in $s$ reduced modulo 2. That is,

$$\text{Par}(s) = \bigoplus_{i=1}^{m} s_i.$$

We will assume that $F$'s message space is the set of all $m$-bit long strings. (We can do without this assumption; see Remark 2 following the protocol.)

**The Coin Flipping Protocol**

(step 1)
$A$ generates, randomly, $n + 1$ keys $(a_0, a_1, a_2, \ldots, a_n)$ to $F$;
$A$ computes $a_{n+i} = a_0 \oplus a_i$, for $1 \le i \le n$;
$A$ chooses, randomly, a message $R_A$ (from $F$'s message space);
$A$ computes $C_A = F_{a_0}(R_A)$ and $C_i^A = F_{a_i}(S)$, $1 \le i \le 2n$;
$A$ transmits $C_A$ and $C_1^A, C_2^A, \ldots, C_{2n}^A$ to $B$;
$B$ acts symmetrically, generating the keys $b_0, b_1, b_2, \ldots, b_n$, picking the message $R_B$ and transmitting its encryption ($C_B$);
(step 2)
$A$ declares that
[Beginning of $A$'s Declaration]
[Denotation:]
The symbols $K_2^A, K_2^A, \ldots, K_{2n}^A$; $K_1^B, K_2^B, \ldots, K_{2n}^B$ denote the solutions of the corresponding $S$-puzzles: $C_1^A, C_2^A, \ldots, C_{2n}^A$; $C_1^B, C_2^B, \ldots, C_{2n}^B$.
The symbols $K_0^A$ and $K_0^B$ denote keys to $F$.
($K_0^A$ must satisfy (1) below, while $K_0^B$ must satisfy (2).)
[Statement:] I am committed to the outcome determined by the evaluation of ''Par$(F_{K_0^A}^{-1}(C_A) \oplus F_{K_0^B}^{-1}(C_B))$'',
[recall: $C_A$ and $C_B$ are values, while $K_0^A$ and $K_0^B$ are symbols!]
if $B$ can present the following (i.e., both (1) and (2)):
(1) Both $K_i^A$ and $K_{n+i}^A$, for some $1 \le i \le n$, so that $K_0^A = K_i^A \oplus K_{n+i}^A$.
(2) $K_j^B$, for all $1 \le j \le 2n$, so that for every $i$, $1 \le i \le n$, $K_0^B = K_i^B \oplus K_{n+i}^B$
[End of $A$'s Declaration]
$A$ signs this declaration and transmits it to $B$;
$B$ acts symmetrically, transmitting $B$'s declaration to $A$;

(Comment: At this stage each has a declaration, signed by his counterpart, that specifies (determines) what will be considered to be the outcome (of the coin-flip).
However, the computation of the outcome is infeasible.)
(step 3)
PSE($A$, $B$, $\{(a_i, a_{n+i})\}_{i=1}^{n}$, $\{(b_i, b_{n+i})\}_{i=1}^{n}$);

**Remarks**

1. Note that, when given only $R_A$, the value of $\mathrm{Par}(R_A \oplus R_B)$ is indistinguishable from an element chosen at random from the set $\{0, 1\}$. The same holds when given $R_B$.

2. Let $\mathcal{M}^{(F)}$ denote the message space of $F$. Let $p_0, p_1, \ldots, p_{t-1}$ be $t$ real numbers whose sum is 1. Let $f$: $\mathcal{M}^{(F)} \times \mathcal{M}^{(F)} \mapsto \{0, 1, \ldots, t-1\}$ be an easy to compute function such that for every $M \in \mathcal{M}^{(F)}$ the following holds:

$$\frac{|\{w \in \mathcal{M}^{(F)}: f(M, w) = i\}|}{|\mathcal{M}^{(F)}|} = \frac{|\{w \in \mathcal{M}^{(F)}: f(w, M) = i\}|}{|\mathcal{M}^{(F)}|} = p_i.$$

Note that using $f(x, y)$ instead of $\mathrm{Par}(x \oplus y)$, one can implement a protocol for conducting a lottery (so that the lottery's outcome is $i$ with probability $p_i$). Also, note that this eliminates the need to assume that $\mathcal{M}^{(F)}$ is the set of all $m$-bit long strings.

## 8. IMPLEMENTATION OF THE 1-OUT-OF-2 OBLIVIOUS TRANSFER

Let $\boxplus$ and $\boxminus$ denote two $\mathcal{M}_x \times \mathcal{M}_x \mapsto \mathcal{M}_x$ operators which satisfy the following:

(i) For every $x$, the mapping $y \mapsto x \boxplus y$ is a permutation on $\mathcal{M}_x$.

(ii) For every $y$, the mapping $x \mapsto x \boxplus y$ is a permutation on $\mathcal{M}_x$.

(iii) For every $x$ and $y$, $(x \boxplus y) \boxminus y = x$.

When using the RSA as the PKCS, it is natural to define $x \boxplus y$ as the reduction modulo $N$ (the RSA's modulus) of $x + y$ and to define $x \boxminus y$ as the reduction modulo $N$ of $x - y$. When using a PKCS the message space of which is the set of all binary vectors of a specific length, it is natural to define both $x \boxplus y$ and $x \boxminus y$ as addition bit-by-bit modulo 2 of the vectors $x$ and $y$.

The following implementation of $\mathrm{OT}_2^1$ was suggested to us by Micali. It is a modification of our original implementation [11] and has several advantages over it.

### The Implementation of $\mathrm{OT}_2^1$

***protocol*** $\mathrm{OT}_2^1(S, R, M_0, M_1)$
(1) $S$ chooses, randomly, one instance of the PKCS, $(E_x, D_x)$;
$S$ chooses, randomly, two messages, $m_0$ and $m_1$, from $\mathcal{M}_x$ (the message space of the above PKCS instance);
$S$ transmits $E_x$, $m_0$, and $m_1$, to $R$;
(2) $R$ chooses, randomly, $r \in \{0, 1\}$;
$R$ chooses, randomly, a message $k \in \mathcal{M}_x$;
$R$ transmits $q = E_x(k) \boxplus m_r$ to $S$;
(3) $S$ computes $k_i' = D_x(q \boxminus m_i)$, for $0 \leq i \leq 1$;
$S$ chooses, randomly, $s \in \{0, 1\}$;
$S$ transmits $(M_0 \boxplus k_s', M_1 \boxplus k_{s\oplus 1}', s)$ to $R$,
(Comment: $\oplus$ denotes addition modulo 2.)

Let us now discuss the validity of the above implementation.

**Discussion**

We assume that the PKCS in use and the $\boxplus$ and $\boxminus$ operators have freeness properties such that: it is infeasible for $R$ to find a $q$ such that he can compute $M_o$ with probability greater than one half; provided $S$ executes the protocol properly.
The following facts are of interest:

1. If both parties follow the protocol properly, then $R$ can read $M_{s\oplus r}$. (Note that $k = k_r'$, that $k$ is known to $R$ and that $M_{s\oplus r} \boxplus k'_{s\oplus(s\oplus r)}$ has been transmitted to $R$.) Thus, axiom (i) of $\mathrm{OT}_2^1$ is satisfied.

2. If $R$ has followed the protocol properly and has not received $M_{s\oplus r}$, then he knows that he has been cheated by $S$. (Notice that $r$ has been chosen by $R$ and that $s$ has been transmitted to $R$.) Thus, $R$ can detect attempts to "cheat" with probability one half. This satisfies axiom (iii) of $\mathrm{OT}_2^1$.

3. If both parties follow the protocol properly, then the only information $S$ gets from $R$ is $q = E_x(k) \boxplus m_r$. Since $k$ is randomly chosen in the message space (and $E_x$ is a permutation operator) this does not give $S$ any information about $r$. Recalling that $r$ and $s$ determine which message will be read by $R$, axiom (ii) follows.

**Remarks**

1. As mentioned in Section 3, OT can be implemented by using $\mathrm{OT}_2^1$. This follows from the trivial reduction: $OT(S, R, M) = OT_2^1(S, R, M, K)$, where $K$ is a-priori known to both $S$ and $R$. Using the above implementation of $\mathrm{OT}_2^1$ it is easy to present (also) a "direct" implementation of OT. One needs only change the transmission of step 3 to the transmission of $(M \boxplus k_s', s)$. Note that the transmission of $s$ in (step 3 of) this implementation of OT is essential, while in $\mathrm{OT}_2^1$ it is not.

2. It is not essential that a user $A$ randomly generates a new instance of PKCS every time he plays the role of $S$ in $\mathrm{OT}_2^1$. $A$ can, just as well, use his publically known instance of the PKCS (i.e., $(E_A, D_A)$); or a special instance he has generated for this purpose.

## APPENDIX

### On the Notion of a Recognizable Secret Message

In order to make the following discussion as simple as possible, we will use intuitive, yet undefined, terms: "feasibility," "negligible," etc. However, these terms can be given a precise meaning by parameterizing the dis-

cussion and defining the terms with respect to a "security parameter," denoted $n_{sec}$. For example, *feasible* is defined as computable within $P(n_{sec})$ time (by some *efficient* algorithm), where $P(\cdot)$ is some polynomial; and a *negligible fraction* is defined as a fraction, $frc(n_{sec})$, such that $frc^{-1}(n_{sec})$ grows faster than any polynomial in $n_{sec}$.

Let $\mathcal{S}$ and $\mathcal{R}$ be two sets and $f$ be a function from $\mathcal{S}$ onto $\mathcal{R}$. We say that $f$ is a *one-way function* if it satisfies the following two conditions:

(i) It is feasible to compute $f(s)$ given $s \in \mathcal{S}$.
(ii) The set of all $r$'s for which it is feasible given $R$ to find an $s$, such that $f(s) = r$, forms a negligible fraction of $\mathcal{R}$.

A message $M$ picked at random from $\mathcal{S}$ is a *recognizable secret* with respect to a one-way function $f$, if $f(M)$ is the only information given about $M$. $M$ is a *recognizable secret for* user $A$ if $f(M)$ is the only information $A$ has about $M$.

***Acknowledgments.*** We are indebted to Silvio Micali for his profound remarks and very useful suggestions. *Mille grazie, Silvio!* We would also like to thank one of the referees for very helpful remarks and Tom Tedrick for reading the manuscript and pointing out several mistakes.

**REFERENCES**

1. Blum, M. private communication, 1981.
2. Blum, M. Coin flipping by telephone. *IEEE Spring COMCON*, 1982.
3. Blum, M. How to exchange (secret) keys. *ACM Trans. Comput. Syst. 1*, 2 (May 1983), 175–193. Also In *Proceedings of the 15th STOC*, 1983, pp. 440–447.
4. Blum, M., and Rabin, M.O. How to send certified electronic mail. in preparation.
5. DeMillo, R., Lynch, N., and Merritt, M. Cryptographic protocols. In *Proceedings of the 14th STOC*, 1982, pp. 383–400.
6. Diffie, W., and Hellman, M.E. New directions in cryptography. *IEEE Trans. Inf. Theory, IT-22*, 6 (Nov. 1976), 644–654.
7. Dolev, D., Even, S., and Karp, R.M. On the security of ping-pong protocols. *Inf. Control 55*, (1982), 57–68.
8. Dolev, D., and Yao, A.C. On the security of public key protocols. In *Proceedings of the 22nd FOCS*, 1981, 350–357. Also in *IEEE Trans. Inf. Theory, IT-29*, 1983, 198–208.
9. Even, S. A protocol for signing contracts. Tech. Rep. 231, Computer Science Dept., Technion, Haifa, Israel, Jan. 1982. Also presented at Crypto 81.
10. Even, S., and Goldreich, O. On the security of multi-party ping-pong protocols. In *Proceedings of the 24th FOCS*, 1983, 34–39.
11. Even, S., Goldreich, O., and Lempel, A. A randomized protocol for signing contracts. Tech. Rep. 233, Computer Science Dept., Technion, Haifa, Israel, Feb. 1982. An extended abstract appears in *Advances in Cryptology: Proceedings of Crypto 82*, D. Chaum, et al. Eds., Plenum Press, New York, 1983, pp. 205–210.
12. Even, S., and Yacovi, Y. Relations among public key signature systems. Tech. Rep. 175, Computer Science Dept., Technion, Haifa, Israel, Mar. 1980.
13. Fischer, M., Micali, S., and Rackoff, C. An oblivious transfer equivalent to factoring. Presented at EuroCrypt 84.
14. Goldreich, O. A protocol for sending certified mail. Tech. Rep. 239, Computer Science Dept., Technion, Haifa, Israel, Apr. 1982.
15. Goldreich, O. On concurrent identification protocols. Tech. Rep. MIT/LCS/TM-250, Massachusetts Institute of Technology, Cambridge, Dec. 1983. Also presented at EuroCrypt 84.
16. Goldreich, O. Sending certified mail using oblivious transfer and a threshold scheme. Tech. Rep. 325, Science Dept., Technion, Haifa, Israel, July 1984. This is a revised version of Appendix H in On the security of cryptographic protocols and cryptosystems. D.Sc. thesis, Computer Science Dept., Technion, Haifa, Israel, 1983.
17. Goldreich, O. A simple protocol for signing contracts. In *Advances in Cryptology: Proceedings of Crypto83*, D. Chaum, Ed., Plenum Press, New York, 1984, pp. 133–136.
18. Goldreich, O., Goldwasser, S., and Micali, S. How to construct random functions. In *Proceedings of the 25th FOCS*, 1984, 464–479.
19. Goldwasser, S., and Micali, S. Probabilistic encryption and how to play mental poker, keeping secret all partial information. In *Proceedings of the 14th STOC*, 1982, 365–377. Also in *J. Comput. Syst. Sci. 28*, 2 (1984), 270–299.
20. Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of theorem-proving procedures. In *Proceedings of the 17th STOC*, to appear.
21. Goldwasser, S., Micali, S., and Rivest, R.L. A paradoxical signature scheme. In *Proceedings of the 25th FOCS*, 1984, 441–448.
22. Hastad, J., and Shamir, A. The cryptographic security of truncated linearly related variables. In *Proceedings of the 17th STOC*, 1985, to appear.
23. Luby, M., Micali, S., and Rackoff, C. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *Proceedings of the 24th FOCS*, 1983, 11–21.
24. Merkle, R.C. Secure communication over insecure channel. *Commun. ACM 21*, 4 (Apr. 1978), 294–299.
25. National Bureau of Standards, Data Encryption Standard, Federal Information Processing Standards, Publ. 46, 1977.
26. Rabin, M.O. Digitalized signatures and public key functions as intractable as factoring. Tech. Rep. MIT/LCS/TR-212, Massachusetts Institute of Technology, Cambridge, 1979.
27. Rabin, M.O. How to exchange secrets by oblivious transfer. unpublished manuscript, 1981.
28. Rabin, M.O. Transaction protection by beacons. Tech. Rep. TR-29-81, Aiken Computation Laboratory, Harvard Univ., Cambridge, Mass., 1981.
29. Rackoff, C., and Luby, M. One-one pseudo-random function generation and DES, in preparation.
30. Rivest, R.L., Shamir, A., and Adleman, L. A method for obtaining digital signature and public key cryptosystems. *Commun. ACM 21*, 2 (Feb. 1978), 120–126.
31. Shamir, A. How to share a secret. *Commun. ACM 22*, 11 (Nov. 1979), 612–613.
32. Tedrick, T. Fair exchange of secrets. In *Proceedings of Crypto84*, to appear.
33. Yao, A.C. Protocols for secure computation. In *Proceedings of the 23rd FOCS*, 1982, 160–164.

**CR Categories and Subject Descriptors:** E.3 **[Data]**: Data Encryption—*data encryption (DES), public key cryptosystems*; H.4.3 **[Information Systems Applications]**: Communication Applications—*electronic mail*; J.1 **[Computer Applications]**: Administrative Data Processing—*financial (e.g., EFTS)*

**General Terms:** Security, Theory

**Additional Key Words and Phrases:** cryptographic protocols, signing contracts, oblivious transfer, public key cryptosystem applications, certified electronic mail, flipping a coin.

Received 2/82; revised 7/84; accepted 10/84

Shimon Even, Department of Computer Science, Duke University, Durham, NC 27706. Oded Goldreich, MIT Laboratory for Computer Science, Cambridge, MA 02139. Abraham Lempel, Computer Science Department, Technion, Haifa 32000, Israel.