Non-Interactive Oblivious Transfer and Applications

Mihir Bellare* Silvio Micali†

MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139

Abstract

We show how to implement oblivious transfer without interaction, through the medium of a public file. As an application we can get non-interactive zero knowledge proofs via the same public file.

1 Introduction

1.1 Non-Interactive Oblivious Transfer

The intriguing concept of an *oblivious transfer* was introduced by Rabin, and has since then proven to be a powerful tool in the design of cryptographic protocols. Interaction, however, has seemed so far to be crucial to any implementation of it. Could one design a non-interactive version of this important primitive? We propose here several ways in which to do this.

The setting we consider is a public key one. Each user B is equipped with a public key P_B and a secret key S_B. A non-interactive oblivious transfer is a means whereby any A can obliviously transfer something to such a B, without the recipient's having to take any action at all. A little more formally,

Non-Interactive Oblivious Transfer : A has two strings s_0 and s_1. As a function of these and B's public key P_B she computes a message m and sends it to B. Using his secret key S_B, B can extract from m exactly one of the strings s_0 or s_1. A will not know which of the two B got.

* Supported in part by NSF grant CCR-87-19689 and DARPA Contract N00014-89-J-1988.
† Supported in part by NSF grant DCR-84-13577, ARO grant DAALO3-86-K-0171, and DARPA Contract N00014-89-J-1988.

A related concept is that of an *oblivious transfer channel.* This is a means of obliviously transferring a lot of information.

Oblivious Transfer Channel (OT channel): An oblivious transfer channel from A to B is a pair $C = (C^0, C^1)$ of channels such that

- A can send any number of bits on either C^0 or C^1
- One of the channels is clear to B (in the sense that he will see any bit that is sent on it) while the other is opaque
- A does not know which channel is clear to B.

OT channels are usually easier to think about and we will see that a single non-interactive oblivious transfer of a pair of short strings can be used to establish these channels.

It should be noted that although an OT channel allows lots of bits to be obliviously transmitted, the obliviousness in not independent. That is, suppose A sends b_0 on C^0 and b_1 on C^1, and suppose B gets b_0. Then, if A now sends c_0 on C^0 and c_1 on C^1, it is c_0 that B will get. Although A does not know which bit of each pair B got, he does know that it is either both b_0 and c_0 or both b_1 and c_1. This can be both an advantage and a drawback. For many applications, though, it is good enough. One such application, which we will describe here, is a construction of multi-user non-interactive zero knowledge systems.

In any case, in a science concerned with secret transmission, a primitive like non-interactive OT remains of fundamental importance and independent interest, over and above the applications visible at this stage.

1.2 Non-Interactive Zero Knowledge

We apply the non-interactive OT to obtain public key non-interactive zero knowledge systems. This is a setup in which there are many users, each with a public key, who can prove theorems to each other in zero knowledge and without interaction. A little more precisely,

Public Key Non-Interactive Zero Knowledge Systems: Consider a community of users, where each user B has a public key P_B and a secret key S_B. We call this a *public key non-interactive zero knowledge system* if for each pair of users A and B, and for each theorem T, it is possible for A to give B a *non-interactive zero knowledge proof* of T. This is a message m which A computes as a function of her theorem and B's public key and which she then sends to B. B's secret key enables him to decode m to the extent that he is satisfied of the correctness of the theorem, but he learns nothing more than that the theorem is true. Note that the communication is in one direction only: in order to receive the proof B need send nothing to A. Moreover, not only can any other user C also send proofs to B, but the number of theorems that can be proved to B is not limited.

Ours are the first implementations of non-interactive zero knowledge proofs which permit many provers and verifiers who do not have to interact individually with one another before proving theorems (non-interactively) in zero knowledge.

Non-interactive zero-knowledge was introduced by Blum, Feldman, and Micali [BFM][1]. They showed how a prover could prove a theorem to a verifier when both parties share a common random string. The drawback of their system, however, was that it was restricted to two parties: if many users wished to prove theorems to each other, each pair of them would have to share a separate random string. This becomes quickly prohibitive as the number of users grows. Their implementation was also somewhat impractical.

Kilian [K1] showed how a theorem could be encoded and then transmitted using oblivious transfer in such a way as to achieve zero knowledge. Kilian, Micali and Ostrovsky [KMO] have a scheme which moves the oblivious transfer to a short pre-processing stage. That is, the prover and verifier first exchange some information via oblivious transfer. This enables the prover, in a later stage, to send the verifier zero knowledge proofs without interaction. The initial interactive phase scheme again means, however, that this scheme is restricted to two parties. On the other hand the encoding of proofs used is quite efficient and the system does not restrict the sizes of theorems.

Our public key zero knowledge systems evolve from the [KMO] work. We replace the initial interactive phase with public keys. To prove theorems we use the same encoding of theorems as [KMO] and accomplish the proofs via non-interactive OT. Since each user either creates his public key himself or gets it from some center (*without* interaction with the person proving theorems to him), anyone can prove theorems to him.

Given that some of our implementations of non-interactive OT are quite efficient, and we use the [KMO] proof encodings, we get some quite efficient implementations of zero knowledge.

Public Key non-interactive knowledge proofs themselves have cryptographic applications; for example, Bellare and Goldwasser [BG] have shown how they can be used for message authentication.

Remark: As pointed out by Crépeau, proofs in [BFM] are transitive (that is, if B received a proof of a theorem from A she could show it to C, and C too would be convinced of the proof). Our proofs are *not* transitive, as is desirable in a zero knowledge proof.

1.3 Results and Organization of this Paper

We begin (§2) with a simple, concrete, and easily implementable scheme for non-interactive oblivious transfer.

The next set of schemes we present (§3) are more theoretical, and involve having a key distribution center. These centers are *not* trusted, and we show appropriate protocols whereby a user can get a key from them and the center gains no information which could compromise the key. These schemes have the advantage of being based on the general assumption of trapdoor permutations.

[1] The implementation described in the original [BFM] paper is not known to have a proof; a correct scheme has been announced by S. Micali [M].

We have relegated to an appendix the description of our principal application: how non-interactive OT can be used to get non-interactive zero knowledge proof systems.

2 Implementing Non-Interactive OT

We describe a simple, concrete implementation of non-interactive oblivious transfer based on the Diffie-Hellman assumption, and then suggest generalizations of this approach to get alternative implementations.

2.1 A Simple Scheme

Fix some prime p and generator g of Z_p^*. Suppose that these, as well as some element C of Z_p^*, are known to all the users in the system, but suppose that nobody knows the discrete log of C (ways of arriving at such situations are discussed later).

The arithmetic in this section will be understood to be mod p; we will write simply g^x rather than $g^x \bmod p$, etc.

How to Get Keys : B picks $i \in \{0,1\}$ at random, $x_i \in \{0,\ldots,p-2\}$ at random, and sets

- $\beta_i = g^{x_i}$
- $\beta_{1-i} = C \cdot (g^{x_i})^{-1}$.

His public key is (β_0, β_1) and his secret key is (i, x_i).

Anyone can check that B's public key (β_0, β_1) is correctly formed by checking that $\beta_0\beta_1 = C$, and before sending him any proofs they will do so. Granted that the discrete log of C is unknown, B cannot know the discrete logs of both β_0 and β_1. Moreover, the public key does not reveal which of the two discrete logs B knows: the pair (β_0, β_1) is randomly distributed over the set of all pairs of elements of Z_p^* whose product is C. This will be crucial to the non-interactive OT we describe below.

The mechanism we use for non-interactive OT is similar to the Diffie-Hellman secret key exchange protocol, and is based on the same complexity assumption:

Diffie-Hellman Assumption : Given g^x and g^y, but neither x nor y, it is hard to compute g^{xy}.

The Diffie-Hellman assumption is one of the oldest and most tried in cryptography.

We can now describe how the non-interactive oblivious transfer of a pair of strings (s_0, s_1) is accomplished:

Non-Interactive OT(s_0, s_1):

In the above notation, let B's public key be (β_0, β_1) and his secret key (i, x_i).

- A picks at random $y_0, y_1 \in \{0,\ldots,p-2\}$ and sends $\alpha_0 = g^{y_0}, \alpha_1 = g^{y_1}$ to B. A then computes $\gamma_0 = \beta_0^{y_0}$ and $\gamma_1 = \beta_1^{y_1}$, and sends $r_0 = s_0 \oplus \gamma_0$ and $r_1 = s_1 \oplus \gamma_1$ to B.

- On receiving α_0 and α_1, B uses his secret key to compute $\alpha_i^{x_i} = \gamma_i$. He then computes $\gamma_i \oplus r_i = s_i$.

B thus receives s_i. The Diffie-Hellman assumption implies that he cannot compute γ_{1-i} (say $\beta_{1-i} = g^{x_{1-i}}$; then $\gamma_{1-i} = g^{x_{1-i}y_{1-i}}$ and B knows $g^{x_{1-i}}$ and $g^{y_{1-i}}$ but neither x_{1-i} nor y_{1-i}). Thus he cannot compute s_{1-i}. A has thus succeeded in obliviously transferring the pair of strings s_0 and s_1. Note that the transfer is indeed non-interactive: B sends nothing to A.

2.2 A More Secure Scheme

The above scheme is simplified as much as possible. In particular, it is not clear exactly how secure s_{1-i} is: the Diffie-Hellman assumption does not say anything about the bit security. This can be fixed by the standard method of using a hard-core bit. The theorem of Goldreich and Levin [GL] implies that predicting the bit $\langle g^{xy}, r\rangle$ given g^x, g^y and random r is as hard as computing g^{xy} given g^x, g^y (where $\langle g^{xy}, r\rangle$ denotes the inner product mod 2 of the strings g^{xy} and r). For a pair of bits (b_0, b_1) we then have

Non-Interactive OT(b_0, b_1):

In the above notation, let B's public key be (β_0, β_1) and his secret key (i, x_i).

- A picks at random $y_0, y_1 \in \{0, \ldots, p-2\}$ and computes $\gamma_0 = \beta_0^{y_0}$ and $\gamma_1 = \beta_1^{y_1}$. She then picks random $r_0, r_1 \in \{0,1\}^k$ (where $k = |p|$) subject to the restriction that $\langle \gamma_0, r_0\rangle = b_0$ and $\langle \gamma_1, r_1\rangle = b_1$. She sends $\alpha_0 = g^{y_0}, \alpha_1 = g^{y_1}$ and r_0, r_1 to B.
- On receiving α_0, α_1 and r_0, r_1, B uses his secret key to compute $\alpha_i^{x_i} = \gamma_i$. He then computes $b_i = \langle \gamma_i, r_i\rangle$.

The Goldreich-Levin theorem together with the Diffie-Hellman assumption imply that b_{1-i} is unpredictable to B.

As an aside, let us also point out that it is easy to modify our implementation of non-interactive OT to obtain a protocol for an *interactive* 1 out of 2 oblivious transfer based on the Diffie-Hellman assumption. Although it was known [GHY],[K2] that oblivious transfer in the interactive framework was possible under this assumption, the implementation arising out of the modification of our non-interactive scheme is simpler and more efficient.

2.3 Non-Interactive OT of More Bits: OT Channels

Suppose that A wishes in fact to oblivious transfer many pairs of strings to B, as would be required, for example, in the applications to non-interactive zero knowledge proofs that we present in Appendix A. She could, of course, just repeat the above as often as is necessary. More efficient, however, might be the following.

A begins by non-interactive oblivious transferring to B a pair of random k bit strings (s_0, s_1). This is done by non-interactively oblivious transferring $((s_0)_j, (s_1)_j)$ for each $j = 1, \ldots, k$ (where $(s_i)_j$ denotes the j-th bit of s_i) via the scheme of §2.2. Using a pseudo-random bit generator G (the hardness of discrete log, which is implied by the Diffie-Hellman assumption, implies the existence of pseudo-random bit generators [BlMi]) she then expands these seeds into the pair of long pseudo-random sequences $G(s_0)$ and $G(s_1)$. To oblivious transfer a pair of strings (r_0, r_1), A sends to B the bitwise X-OR of r_0 with the next unused bits of $G(s_0)$ and the bitwise X-OR of r_1 with the next unused bits of $G(s_1)$. B gets r_i since he knows the seed s_i, but gets no information about r_{1-i} since without the knowledge of s_{1-i} the sequence $G(s_{1-i})$ looks random to him.

More formally, the above establishes an OT channel. The method used is a general one.

Once OT channels are available, we can implement non-interactive zero knowledge proof systems via the methods outlined in Appendix A.

2.4 2 Out of 3 Non-Interactive OT

A particularly interesting variant of OT is the 2 out of 3 OT. Here A has three bits (b_0, b_1, b_2). B selects two of them and A does not know which pair of bits B got. The above scheme for 1 out of 2 non-interactive OT can easily be modified to directly implement a non-interactive 2 out of 3 OT. B will make his public keys as follows:

B picks at random a pair of distinct values $i, j \in \{0, 1, 2\}$, and then picks at random $x_i, x_j \in \{0, \ldots, p-2\}$. He sets

- $\beta_i = g^{x_i}, \beta_j = g^{x_j}$
- $\beta_l = C \cdot (g^{x_i})^{-1} (g^{x_j})^{-1}$, where $l \in \{0, 1, 2\}$ is the value not equal to i or j.

His public key is $(\beta_0, \beta_1, \beta_2)$ and his secret key is (i, j, x_i, x_j).

It is then easy to see how to generalize the scheme of §2.2 to define a **Non-Interactive 2 Out of 3 OT**(b_0, b_1, b_2), and we omit the details.

The interest of this variant of OT lies in its application to zero knowledge proofs via the results of [KMO]. They show a simple, efficient, general, and *non-cryptographic*[2] method of "zero knowledge proofs for NP in three envelopes" which can be used to directly implement non-interactive zero knowledge proofs via our non-interactive 2 out of 3 OT.

We note that the scheme described here is easily generalized to achieve a $t-1$ out of t non-interactive OT for any t.

2.5 The Central Public Key

The above schemes requires the presence of some short string, common to all parties and satisfying some constraints, which can be used by any user to create his public

[2] The two envelope scheme described in Appendix A is cryptographic in the sense that creating the envelopes requires using encryption functions.

and secret keys. We might call this a *central public key*. Specifically, the central public key in the above consists of a random string C whose discrete log nobody know, together with a prime p and a generator g of Z_p^* for which the discrete log problem is hard.

How can a central public key with the desired properties be obtained? The simplest and most direct way would be to have a center create it. Its job done, the center could disappear. This would probably work well enough in practice.

If one does not want a center then multi-party protocols as in [GMW],[BGW], or [CCD] could be used by the users themselves to agree on a central public key. These protocols have the necessary feature of not allowing any user (or any small subset of users) to influence the choice of the key to their advantage.

2.6 A Proof of Security

In order to formally prove that the oblivious transfer has the right properties, we will have to add one more step. When B makes his public key, we have him publish a zero knowledge proof that he really did it correctly (formally, that he knows i, x_i such that $\beta_i = g^{x_i}$). In the simulation the simulator will use this proof to extract the value of i.

Such a proof could be implemented via [BFM]. Unfortunately the scheme of [BFM] is based on quadratic residuosity. In the final paper we will show how to get some kind of proof based on discrete log, at the expense of a small interaction with the center.

2.7 Other Implementations

In the final paper we will consider a general framework which encompasses schemes of the above sort. The idea is that a user should be able to create his public key on his own, using some central public key. Moreover, there is a pair of secrets associated with his public key of which he only knows one. This is guaranteed by the fact of some relation between his public key and the central key being true, and this relation can be checked by anyone. Given this, there is a way to establish two encryption algorithms only one of which the key holder can decrypt. These are used for the non-interactive OT.

3 Schemes with Centers

The simplest and most direct way in which to establish public and secret keys which permit non-interactive OT would be through the use of a key distribution center. For example, consider a center who gives B two numbers N_0 and N_1, only one of which is given in factored form. B makes (N_0, N_1) his public key. To non-interactively oblivious transfer things to B, A can use these numbers to establish encryption algorithms which are used to send the bits in encrypted form. B can only decrypt those bits sent using the number whose factorization he knows.

A closer examination of this idea shows that some care must be exercised. There are a variety of drawbacks to the naive use of centers:

- The center knows which channel is clear to B (he knows which number B has the factorization of). If he reveals this to A the latter can cheat.
- B might disregard what the center gives him and simply create, on his own, a key in which he places a pair of numbers *both* of whose factorizations he knows. He now extracts knowledge from the proofs he receives.

We propose here a way in which a key distribution center can be used to get appropriate keys while avoiding drawbacks of the above form. We will guarantee that after B gets a key from the center,

(1) The center does not know which channel is clear to B

(2) B cannot change his key, or use another key which he builds to suit himself.

We will do this using oblivious circuit evaluation, trapdoor permutations, and digital signatures.

Oblivious Circuit Evaluation : K has some input x and B has some input y. Both know a function F. At the end of the protocol the following holds:

- B learns the value of $F(x, y)$
- B learns no more information about x than that conveyed by the knowledge of $F(x, y)$
- K learns nothing about y or $F(x, y)$.

Oblivious circuit evaluation is a well known protocol of which numerous implementations exist. In particular it can certainly be done given the existence of trapdoor permutations.

Let G be a trapdoor permutation generator (that is, G is a probabilistic polynomial time algorithm which can be used to produce a random trapdoor permutation together with its inverse). Suppose that the center K has a public key P_K with respect to which it can provide signatures; secure digital signatures with trapdoor permutations are possible via [BeMi]. In order to compute the signatures K also has a secret key S_K.

The circuit F we consider takes as input a secret key S and a public key P for signatures, a bit i, and two strings r and s. The output is

$$F(S, P, r, s, i) = ((f_0, f_1), \sigma, f_i^{-1}) ,$$

where f_0 and f_1 are trapdoor permutations, f_i^{-1} is the inverse to f_i, and σ is a signature, with respect to the public key P, of the pair (f_0, f_1). To create these trapdoor permutations, F runs G twice, using as coin tosses the string $r \oplus s$. From the two pairs $(f_0, f_0^{-1}), (f_1, f_1^{-1})$ so obtained, F outputs (f_0, f_1) and a signature σ, with respect to P, of the string (f_0, f_1). The latter is created using P and S. F then also outputs f_i^{-1}.

When user B wishes to get keys, he engages in an oblivious circuit evaluation protocol for F with the center K. B provides the inputs s and i which he chooses at random. K provides r, which he chooses at random, and his own keys $P = P_K$

and $S = S_K$ for signatures. The output goes to B. He makes $((f_0, f_1), \sigma)$ his public key and f_i^{-1} his secret key. The two properties listed above do hold: (1) K does not learn i and hence does not know which channel will be clear to B (2) B cannot make a key to suit himself because he would not be able to produce a signature of it with respect to K's public key (the strong properties of the digital signatures of [BeMi] insure that this latter remains true even after B has seen many signatures from the center).

We note that we cannot, of course, protect against a totally corrupted center: for example, one who is willing to conspire with B and sign for him a bogus public key not obtained through the oblivious circuit evaluation.

To implement non-interactive OT with keys of the form B obtains here is easy: the trapdoor permutations can be used by A to send B encrypted bits. B can decode only one of these streams of bits since he knows only one of the trapdoors. The distribution according to which B's public key is chosen is such that A cannot tell which trapdoor permutation it is that B knows the inverse of.

4 The Non-Interactive OT Primitive

One of the features of the usual interactive OT which makes it a tool of such universal application in the design of interactive protocols is that many stronger versions of OT can be reduced to the simplest kind. Such reductions appear in the work of Brassard, Crépeau, and Robert [BCR]. The same holds true for non-interactive OT. In several cases the reductions of [BCR] apply since they do not involve interaction over and above that of the original protocol. It is interesting to note, however, the case of the most interesting reduction: how a 1 out of n bit transfer yields a 1 out of n string transfer. For the non-interactive case, the reduction is actually much *simpler* than the one for the interactive case.

A Appendix: Non-Interactive Zero Knowledge Proofs

This appendix describes how non-interactive zero knowledge proofs are accomplished via OT channels. For a definition of OT channels see §1.1.

Kilian, Micali and Ostrovsky [KMO] presented a method via which theorems could be proved, non-interactively and in zero-knowledge, based on obliviously transferred pseudo-random sequences; two seeds would be known to the prover, and only one of them (with the prover unaware of which one) to the verifier. Their scheme will work in the more general framework of oblivious transfer channels, and we describe it in that form here.

Suppose A wishes to prove some NP statement T of which she knows a witness. She transforms this to a graph $G = (V, E)$ of which she knows a Hamiltonian cycle, where $V = \{1, \ldots, n\}$. Let A_G be the adjacency matrix of G and let the edges of the Hamiltonian cycle be $(u_1, v_1), \ldots, (u_n, v_n) \in E$. Let k be a security parameter. We assume that k OT channels $C_1 = (C_1^0, C_1^1), \ldots, C_k = (C_k^0, C_k^1)$ from A to B are available (k being some security parameter).

A picks a random permutation π of the vertices of G and computes $G' = \pi(G) = (V, \pi(E))$, the isomorphic image of G under π. Let $A_{G'} = [a']_{ij}$ be the adjacency matrix of G'. She then picks some encryption function $\mathcal{E}(\cdot,\cdot)$[3], and does the following:

- Choosing r at random she encrypts π as $y = \mathcal{E}(\pi, r)$.
- She encrypts the adjacency matrix of the permuted graph by choosing r_{ij} at random and computing $y_{ij} = \mathcal{E}(a'_{ij}, r_{ij})$ for each $i, j = 1, \ldots, n$.

She now sends y and y_{ij} $(1 \leq i, j \leq n)$ to B in the clear.

Finally, we arrive at the point where A uses an OT channel. She flips a coin. If it is heads, she sends r, r_{ij} $(1 \leq i, j \leq n)$ along C_1^0 and $r_{\pi(u_1)\pi(v_1)}, \ldots, r_{\pi(u_n)\pi(v_n)}$ along C_1^1. If the coin is tails she reverses the roles of C_1^0 and C_1^1 in the above.

Actually, A repeats this entire procedure k times. That is, she obtains k encodings as described, and she uses C_i to transfer the i-th encoding to B.

To prove a further theorem, A does the same thing. She uses the same set of k channels (recall that they can take an unlimited number of bits).

At the receiving end, B sees either the permutation and the permuted graph, or a Hamiltonian cycle in a permuted version of the graph. This is a well known zero-knowledge proof of Hamiltonian cycle. A full proof of correctness, however, would require showing a simulator and a reduction via which the ability to distinguish the simulator's output from the prover's would compromise either the encryption function or the channels. Details of this sort are left to the final paper.

[3] This is [GM] style probabilistic encryption: to encrypt a string x, choose a random r and compute $\mathcal{E}(x, r)$; to decrypt, reveal r.

References

[BG] Bellare, M., and S. Goldwasser, "A New Paradigm for Digital Signataures and Message Authentication based on Non-Interactive Zero Knowledge Proofs," CRYPTO 89.

[BeMi] Bellare, M., and S. Micali, "How to Sign Given Any Trapdoor Function," STOC 88.

[BGW] Ben-Or, M., S. Goldwasser, and A. Wigderson, "Completeness Theorem for Non-Cryptographic Fault Tolerant Distributed Computing," STOC 88.

[BlMi] Blum, M., and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," *SIAM Journal on Computing*, Vol. 13, No. 4 (November 1984), 850-864.

[BCR] Brassard, G., C. Crépeau, and J.-M. Robert, "Information Theoretic Reductions among Disclosure Problems," FOCS 86.

[BFM] Blum, M., P. Feldman and S. Micali, "Non-Interactive Zero Knowledge and its Applications," STOC 88.

[CCD] Chaum, D., C. Crépeau and I. Damgård, "Multiparty Unconditionally Secure Protocols," STOC 88.

[GHY] Galil, Z., S. Haber and M. Yung, "Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public Key Model," CRYPTO 87.

[GM] Goldwasser, S., and S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences* 28 (April 1984), 270-299.

[GL] Goldreich, O. and L. Levin, "A Hard-Core Predicate for any One-Way Function," STOC 89.

[GMW] Goldreich, O., S. Micali and A. Wigderson, "A Completeness Theorem for Protocols with Honest Majority," STOC 87.

[K1] Kilian, J., "Founding Cryptography on Oblivious Transfer," STOC 88.

[K2] Kilian, J., personal communication.

[KMO] Kilian, J., S. Micali and R. Ostrovsky, "Efficient Zero Knowledge Proofs with Bounded Interaction," CRYPTO 89.

[M] Micali, S., personal communication, March 1989.

Session 12:

Multiparty computation

Chair: KEVIN MCCURLEY