Lecture 22 - Oblivious Transfer (OT) and Private Information Retrieval (PIR)

Boaz Barak

December 8, 2005

Oblivious Transfer We are thinking of the following situation: we have a server and a client (or a sender and a receiver) where the server has a list of n strings $x_1, \ldots, x_n$, and the client wants to learn x_i. Of course the client can simply send i to the server, but the client does not want the server to know i. Now, the server can simply send the n strings to the client but the server does not want the client to learn x_j for $j \neq i$.

This problem, originating with Rabin, where the server should transfer x_i to the client without knowing i, is called *oblivious transfer*.

Possible application Suppose that we want to an electronic-cash scheme. The idea is that Alice will set up a service, where users like Bob can purchase from her "digital vouchers" that they can later use to buy things from vendors. The vendors will then present these vouchers to Alice.

One naive solution will be the following: Alice will prepare the message $m =$ ``I am willing to pay \$1'' and a signature σ on m. Whenever Bob pays her \$$x$ Alice will give Bob x copies of the message m and the signature σ. Vendors can test the signature when Bob pays for something, and whenever Alice is presented with (m, σ) she will pay the vendor \$1. This solution is simple but it has the drawback that it will probably not take Bob or the vendors long to find out that they can copy m and "print money".

Now suppose that Alice tries a new solution: she has a counter id which is initially set to 1, and whenever Bob pays her \$1, Alice provides bob with a signature on $m_{id} =$ ``I am willing to pay \$1. ID= id'' and then increment id by one. Now she keeps a database of all the vouchers she cached, so she will never cash the same voucher twice. Furthermore, she makes this database available online so that vendors can verify that a voucher is valid before accepting it.

This is great, but now Bob is not that happy. The only reason he wanted to use e-cash and not his credit card was that he did not want his transactions to be traceable, but now Alice knows exactly what id he was given, and so knows exactly what he bought with e-cash (which can be quite embarrassing for Bob - for example he may have bought a cryptography book that uses the random oracle model).

To solve this problem and make Bob happy but Alice not bankrupt, we use oblivious transfer. Alice prepares in advance messages $m_1, \ldots, m_n$ (where n is the amount of money she has) and corresponding signatures $\sigma_1, \ldots, \sigma_n$. She lets $x_i = m_i \circ \sigma_i$. When Bob pays her \$1, they run an oblivious transfer protocol with Alice's inputs being $x_1, \ldots, x_n$ and Bob choosing i at random. At the end of the protocol, Bob got exactly one valid pair m_i, σ_i (if it is not

valid, Bob can reveal his private coins and prove that he deserves to get his money back), and Alice does not know i. Again, when Alice is presented with a message m_i, σ_i she cancels this voucher, removes it from the list used in future oblivious transfers, and makes the list of canceled i's public.[1]

Private Information Retrieval In the previous example n is the amount of money Alice has, which can be quite a lot sometimes. Now, if we assume that the length of a signature is k, it can be that k is much smaller than n, and since Bob is only interested in learning $O(k)$ bits, it would be great if the communication in the protocol could be only a function of k, and be much smaller than n. Such a protocol is called a *private information retrieval* (PIR) protocol. Constructing such protocols with $\ll n$ communication is non-trivial even if we don't care about the security of the server (and so the standard definition does not require this).

Side note: Multiple server PIR: Even without considering security for the server, it can be shown that without considering computational security, we cannot have a PIR protocol with $o(n)$ communication. However, it turns out that there is in fact a model where we can have *information theoretic* security for PIR. This is the model where there are several servers that all hold the same database and do not communication with one another. The verifier will ask all the servers some queries, such that each server will have no information about the index i (even though i can be reconstructed from all the queries together). It was shown by Beimel, Ishai, Kushilevitz and Raymond how to construct such a protocol for c servers and communication roughly $n^{1/c}$ (actually it's slightly better - it's something like $n^{\log\log c/(c\log c)}$ this difference is important since people conjectured for some time that $n^{1/c}$ is the right bound) . It is not known whether this is the best possible - as far as we know it may be even possible to have a protocol with polylogarithmic communication. This question is tightly related to a question in coding theory about the existence of *locally decodable codes* with good parameters.

Secure function evaluation Note also in that example that Alice did not really need to store the entire database $x_1, \ldots, x_n$. Rather she had an efficient algorithm that on input i, outputs the pair m_i, σ_i, where this algorithm runs in time polynomial in the length of i (which is equal to $\log n$) and the output k. Thus, we might be interested in a protocol for OT where both the communication and the running time of both parties is polynomial in $|i|$ and k. Such a protocol is called a protocol for *secure function evaluation* (SFE), since it helps two parties compute the function $f(s, i) = m_i \circ \sigma_i$, where s is the server's private input (signing key) and i is the index. We allow the communication and computation of secure function evaluation can depend on the time to compute the i^{th} entry in the list. Thus, if we have an SFE protocol for every function then we obtain immediately an OT protocol, but this is incomparable with a PIR protocol. (Verify that you understand why.)

Results The following results are known:

- Secure function evaluation can be constructed using oblivious transfer.
- Oblivious transfer can be constructed using one-out-of-two oblivious transfer (the case $n = 2$).

[1] There is a slight chance that Bob will accidentally pick i that was already given previously to Charlie. For simplicity we assume that in this case Bob loses his money, although he can make the probability of this happening very low by choosing i at random. We can think of this probability as an added transaction cost for the service (although it's not clear Bob will see it this way).

- One-out-of-two oblivious transfer can be constructed based on any trapdoor permutation (see Goldreich's book Vol II), but also more efficiently based on specific number theoretic assumptions. The technique used is to first construct a protocol for the *honest but curious model* (essentially a passive adversary that gets to read the private information of one of the parties, but not modify its behavior) and then transform it into a protocol for the general case using zero knowledge. The construction we'll show is a simpler one by Naor and Pinkas based on the DDH.
- Secure function evaluation can be constructed based on oblivious transfer (see Goldreich's book).
- Private information retrieval (single server computational version) can be constructed with n^ϵ communication based on hardness of deciding quadratic residuosity (Kushilevitz and Ostrovsky, we'll show a simpler version of this protocol with communication $n^{1/2+\epsilon}$), and polylogarithmic communication based on somewhat more exotic assumptions [Stern 98], [Cachin, Micali, Stadler '99], [Chang '04], [Limpaa '04].

OT based on the DDH. We will show an OT protocol for the case that $n = 2$. We will then use such a protocol to construct an OT protocol for general n.

Preliminaries We'll use the following lemma, which was proven in the secret sharing lecture:

Lemma 1. *Let q be a prime and let $\mathbb{F}_q$ equal the set $\{0, \ldots, q-1\}$ with addition and multiplication modulu q. Let $a \neq a'$ be two elements in $\mathbb{F}_q$. Then for a random degree 1 polynomial $f(x) = rx + s$ (where r, s are chosen at random in $\mathbb{F}_q$), the random variable $\langle f(a), f(a') \rangle$ is distributed according to the uniform distribution over $\mathbb{F}_q \times \mathbb{F}_q$.*

The protocol We now turn to describing the protocol. As usual in discrete-log based protocols, it will be convenient for us to give variable names for the discrete logs of the numbers involved, but keep in mind that a party given g^a cannot compute a. (This description is actually missing a crucial step, see below.)

Protocol OT_1^2:

Sender's input $x_1, x_2 \in \{0,1\}^*$.

Receiver's input $i \in \{1, 2\}$.

Receiver's message Choose at random prime p of the form $p = 2q+1$ for prime q, let G be the group of quadratic residues modulu p, (note that $|G| = q$) and g be a generator for this group. Receiver chooses a, b at random in $\{0, \ldots, p-1\}$. It then chooses $c_i = a \cdot b \pmod q$ and $c_{\overline{i}}$ is chosen at random from $\{0, \ldots, p-1\}$ (where $\overline{1} = 2$ and $\overline{2} = 1$). It sends to sender the primes p, q, the generator g, and the tuple $\langle g^a, g^b, g^{c_1}, g^{c_2} \rangle$.

Sender's message Sender is given a tuple $\langle g^a, g^b, g^{c_1}, g^{c_2} \rangle$ (although note that it does *not* know a, b, c_1, c_2). It verifies that all of these belong to the group G and that $g^{c_1} \neq g^{c_2}$. It then chooses at random $r, s \in \{0, \ldots, q-1\}$, and sends to the receiver $w = (g^a)^r g^s$ and z_1, z_2 where for $i = 1, 2$, $z_i = \pi_i \oplus x_i$ and

$$\pi_i = (g^{c_i})^r \left(g^b\right)^s$$

(Note that we assume here that a random element of the group G is represented by a random string in $\{0,1\}^\ell$ for some ℓ. We can get the same effect by encoding x_i as an element in G and use multiplication in the group instead of $\oplus$ to mask it.)

Receiver's computation Receiver got three strings w, z_1, z_2 from the sender. It computes $\pi_i = w^b$ and $x_i = z_i \oplus \pi_i$ as its output.

Analysis The security for the receiver is based on the DDH assumption. The security for the prover actually holds information theoretically, regardless of the computational powers of the receiver. This is proven in the following lemmas:

Lemma 2 (Receiver's security)**.** *Assume DDH is true. Then the receiver's message when $i = 1$ is computationally indistinguishable from the receiver's message when $i = 2$.*

This is almost immediate and is left as an exercise.

Lemma 3 (Sender's security)**.** *If $c_i = a \cdot b$ for some $i \in \{1, 2\}$, then that $\pi_{i'}$ (for $i' = \bar{i}$) is a uniform element of the group G, even conditioned on all the rest of the information the sender provides.*

Note that this does indeed imply that $x_{i'}$ is completely hidden from the receiver.

Proof. The main part of the receiver's message consists of a four-tuple $g^a, g^b, g^{c_1}, g^{c_2}$. Since the prover verifies that $g^{c_1} \neq g^{c_2}$ we know that $c_1 \neq c_2$. Therefore there exists i' such that $c_{i'} \neq a \cdot b \pmod{q}$.

The prover selects random r and s and then computes $w = g^{ar+s} = g^{f(a)}$ where $f(x) = rx+s$. The number π_i is equal to $g^{abr+bs} = g^{bf(a)}$. The number $\pi_{i'}$ is equal to $g^{c_{i'}r+bs} = g^{b(r \cdot c_{i'}/b+s)} = g^{b \cdot f(c_{i'}/b)}$. Now the only parts of the sender's message that depend on s and r are the values $w = g^{f(a)}$ and $z_i = g^{bf(a)} \oplus x_i$. Since these depend only on $f(a)$, even conditioned on these values, the value $f(a')$ for $a' = c_{i'}/b$ is completely random. □

To make this into a secure OT protocol that remains secure even when the receiver can deviate from the protocol, we can add an intermediate step where the receiver will prove in zero-knowledge that $c_i = a \cdot b$ for some i.

Getting to OT_1^n Given such a protocol, we can convert into a protocol for choosing one input out of n in the following way:

Sender's input $x_1, \ldots, x_n \in \{0,1\}^\ell$.

Receiver's input $i \in \{1, \ldots, n\}$.

Operation Sender chooses $k_0 = 0^\ell$. For $j = 1, \ldots, n$ the protocol proceeds in the following way:

- Sender chooses k_j at random from $\{0,1\}^\ell$.
- Sender and receiver run 1-out-2 OT with sender's first input equalling $k_0 \oplus \cdots \oplus k_{j-1} \oplus x_j$ and sender's second input equalling k_j. If $i = j$ then the receiver asks to receive the first input, and otherwise it asks to receive the second.

Receiver's computation The receiver learned k_j for all $j \neq i$ and $k_0 \oplus \ldots \oplus k_{i-1} \oplus x_i$ and so it can recover i.

Analysis Security for the receiver follows fairly immediately from security of the basic 1-out-2 OT. For the sender's security note that in each iteration the receiver learns only one input. Let i denote the first iteration in which this input is the first one. (If the receiver learns the second one in all the iterations then it received no information about the inputs.) For $j < i$, the receiver didn't receive any information about x_j. For $j > i$ the receiver can at best receive x_j XOR'd with some things that include also k_i (which he does not know) and so will also receive no information about x_j.

The PIR protocol We sketch the PIR protocol of Kushilevits and Ostrovsky. It has $O(\sqrt{n}k)$ communication where n is the number of elements and k is the security parameter. (Recall that we are now thinking of only the verifier's security.)

Quadratic Residuosity assumption Recall that if factoring is hard then extracting square roots modulu a composite is hard. Similar to the DDH, we'll now make a stronger assumption that it is in fact hard do distinguish between a random quadratic residue and a random element of $\mathbb{Z}_m^*$ (we use m since n is taken in this context by the number of elements). Again, similar to DDH, this does not make sense as is and we have to move to a subgroup of $\mathbb{Z}_m^*$ (which we denote by Z_m^+). This subgroup contains all the quadratic residues and membership in it is polynomially testable.[2] That is, we make the following assumption: for a random $m = p \cdot q$, it is hard to distinguish between a random element of QR_m and a random element of $\mathbb{Z}_m^+$. Note that given the factorization of n it is easy to find out whether or not a number y is a quadratic residue.

The protocol We will design a protocol for the case that each of the sender's inputs is just one bit (we can run the protocol several times one after the other to obtain more bits). We think of this as a $\sqrt{n}$ by $\sqrt{n}$ matrix x, where $x_{i,j}$ is the bit at the place i, j. The receiver wants to find out x_{i_0,j_0} for some i_0, j_0.

Receiver's message Choose $m = p \cdot q$ for random p, q. Choose $y_1, \ldots, y_{\sqrt{n}}$ in the following way: if $j \neq j_0$ then y_j is a random quadratic residue modulu m. y_{j_0} is chosen to be a random quadratic non-residue modulu m. Receiver sends $y_1, \ldots, y_{\sqrt{n}}$.

Sender's message For every row i in the matrix, the sender computes the number z_i as follows: z_i equals the product of all the y_j's for every j such that $x_{i,j} = 1$. The sender sends $z_1, \ldots, z_{\sqrt{n}}$ to the receiver.

Receiver's computation If z_{i_0} is a quadratic residue then the receiver decides that $x_{i_0,j_0} = 0$ and if z_{i_0} is a quadratic non-residue then the receiver decides that $x_{i_0,j_0} = 1$

See the paper of Kushilevitz and Ostrovsky for the analysis.

Reducing the communication The idea of reducing the communication further is that instead of the sender sending to the receiver $z_1, \ldots, z_{\sqrt{n}}$ the receiver can run the PIR protocol recursively to obtain z_{i_0}. Then, by choosing a matrix with fewer columns, it is possible to save on communication and get a more efficient protocol.

[2] Technically this is the subgroup of elements with Jacobi symbol equalling $+1$. For $m = pq$, the group of quadratic residues has size $|\mathbb{Z}_m^*|/4$ while $\mathbb{Z}_m^+$ has size $|\mathbb{Z}_m^*|/2$.