

Advanced Topics in Cryptography

Lecture 1 Benny Pinkas

Administrative Details

- Grade
 - Based on homeworks (might include a final take home exam)
- Office hours: Wednesday, 12-13.
- Email: benny@cs.haifa.ac.il
- Web page:
<http://www.pinkas.net/courses/atc/2006/index.html>
- Goal: Learn a selection of advanced topics in cryptography

Course Outline

- Course Outline
 - Cryptography is more than just encryption/signatures:
 - Oblivious transfer, secure computation.
 - Privacy preserving database operations: Private information retrieval (PIR), computing intersections, keyword search.
 - Search on encrypted data.
 - Advanced topics in encryption:
 - Encryption with chosen-ciphertext security.
 - Homomorphic encryption.
 - Pairing based cryptography. Identity based encryption (IBE). Signature schemes.
 - Broadcast encryption.
 - Side channel attacks.

Today Lecture: Oblivious Transfer

- Web resources:
 - Ronald Cramer's lecture notes,
http://homepages.cwi.nl/~cramer/papers/CRAMER_revised.ps.
 - Boaz Barak's lecture
<http://www.cs.princeton.edu/courses/archive/fall05/cos433/lec22.pdf>
 - Ben Lynn's page
<http://rooster.stanford.edu/~ben/crypto/ot.php>
 - Naor, Pinkas, "Computationally Secure Oblivious Transfer"
<http://www.springerlink.com/index/10.1007/s00145-004-0102-6>

An application: computing “AND” privately

- Two players, P1 and P2, have *binary* inputs a and b .
- They wish to compute $a \text{ AND } b$ without revealing any other information about their inputs.
- If P1's input is $a=0$, and he learns that $(a \text{ AND } b) = 0$, he does not learn whether P2's input is 0 or 1.
- This is the first example of “secure computation” that we will learn
- Applications?

1-out-of-2 Oblivious Transfer

- Two players: sender and receiver.
 - Sender has two inputs, x_0, x_1 .
 - Receiver has an input $j \in \{0,1\}$.
- Output:
 - Receiver learns x_j and nothing else.
 - Sender learns nothing about j .
- Depending on the OT variant, the inputs x_0, x_1 could be strings or bits.
- Suppose that we have a construction of bit OT, how can we solve the AND computation problem?

Computing “AND” Privately using OT

- P1 plays the sender’s part, P2 is the receiver.
 - P1 sets the sender’s inputs to be $x_0=0$, $x_1=a$.
 - P2 sets the receiver’s input to be $j=b$.
- They run an OT protocol, and P2 sends the final answer to P1.
- The output is $(1-j) \cdot x_0 + j \cdot x_1 = (1-b) \cdot 0 + b \cdot a = a \cdot b$.
- Privacy (hand-waving):
 - If $b=0$ then the result that P2 obtains in the OT protocol is always 0 and does not reveal anything about a .
 - If $a=0$ then the result obtained in the OT protocol is always 0. When it is forwarded to P1 it does not reveal anything about b .

Computing “AND” Privately using OT

- Privacy:
 - We show that P2 does not learn more than the “AND” result. We do that by showing that given the output of the “AND” function, it is possible to provide P2 with its “view” in the “AND” protocol (namely, the output of the OT).
 - Proof: we receive the output of the “AND” function and construct the input of the server (P1) in the OT protocol. Therefore P2’s output in the OT protocol is a function of the “AND” result.
 - input: $(a \text{ AND } b)$
 - We set $x_0=0$, $x_1=a \text{ AND } b$.
 - The output of the protocol is as in the original protocol.
 - How can we show a similar argument showing that P1 does not learn more than the “AND” result?

Security definitions

- We must first specify the power of the adversaries.
- Semi-honest vs. malicious adversaries:
 - Semi-honest (honest but curious) adversaries follow the protocol, but might try to learn additional information from the messages they receive during protocol execution.
 - Malicious adversaries might behave arbitrarily.
 - For instance, if the protocol requires P1 to send to P2 a random number z , a malicious P1 might define z by choosing a random y and defining $z = \text{Enc}(y)$.
- We can now define what it means for an OT protocol to be secure. These definitions can be applied to both semi-honest and malicious adversaries.

Security Definitions

- Now we must state what it means not to learn anything more than the output of the protocol.
- For the receiver's security, this is easy, since the sender receives no output.
- As for sender's security, this is more tricky.

Security Definitions

- Receiver's security – indistinguishability
 - For any values of x_0, x_1 , the sender cannot distinguish between the case that the receiver's input is 0 and the case that it is 1.
- Sender's security – comparison with the ideal model
 - The receiver should only learn the defined output, as if there was an ideal implementation of OT.
 - Ideal implementation:
 - There is a third trusted party (TTP).
 - The TTP receives x_0, x_1 from the sender, and j from the receiver. It sends x_j to the receiver.

Security Definitions

- Definition of sender's security:
 - For every algorithm A' that the receiver might run in the real implementation of oblivious transfer
 - there is an algorithm A'' that the receiver can run in the ideal implementation
 - such that for any values of x_0, x_1 the outputs of A' and A'' are indistinguishable.
 - Namely, the receiver in the real implementation does not learn anything more than the receiver in the ideal implementation.
- These definitions do not handle delicate issues, such as
 - whether the receiver “knows” j or the sender “knows” x_0, x_1
 - Whether the parties are committed to their inputs

Constructions of OT

- There is no OT protocol which provides unconditional security for both parties.
- Namely, information theoretic security which does not depend on any computation assumption (just like a one-time-pad).
- We show this by showing that there is no AND protocol which provides unconditional security for both parties

Impossibility of achieving unconditional security

- Suppose that there is an AND protocol (Between P1 and P2, with inputs a and b) with unconditional security
 - Let T be a transcript of all messages sent in the protocol.
 - The parties use random inputs R_1 and R_2 , and given these inputs the transcript T is a deterministic function.
 - Suppose that $a=0$.
 - If $b=0$, then there exists a R'_1 s.t. if P1 has inputs $a=1$ and R'_1 , the protocol would have produced the same transcript T .
 - If $b=1$, then there cannot be any R''_1 s.t. if P1 has inputs $a=1$ and R''_1 the protocol results in the transcript T (this is because the output of the protocol in this case is “1”).
 - P1 can search over all possible values for R_1 and check if running the protocol with $a=1$ and any of them results in T . If there is such an R_1 then it concludes that $b=0$.

The Even-Goldreich-Lempel 1-out-of-2 OT construction

- Setting:
 - Sender has two inputs, x_0, x_1 .
 - Receiver has an input $j \in \{0, 1\}$.
- Protocol:
 - Receiver chooses a random public/private key pair (E, D) .
 - It sets $PK_j = E$, and chooses PK_{1-j} at random from the same distribution as that of public keys*. It then sends (PK_0, PK_1) to the sender.
 - The sender encrypts x_0 with PK_0 , and x_1 with PK_1 , and sends the results to the receiver.
 - The receiver decrypts x_j .
 - Why is this secure against semi-honest adversaries?
- (*) It is required that it is possible to sample items with the exact distribution of public keys, and do this without knowing how to decrypt the resulting ciphertexts.

The Bellare-Micali Construction

- Preliminaries:
 - G_q is a subgroup of order q of Z_p^* , where p is prime and $p=2q+1$.
 - The OT protocol is secure assuming that the Diffie-Hellman assumption holds for G_q .
 - The Computational Diffie-Hellman assumption (CDH) is that the following problem is hard: The input contains a generator g and values g^a, g^b generated with random $a, b \in [1, q]$. The task is to find $z = g^{a \cdot b}$.
 - Recall also the Decisional Diffie-Hellman problem: The input contains a generator g and values g^a, g^b generated with random $a, b \in [1, q]$; and a pair (z, z') where one of (z, z') is $g^{a \cdot b}$ and the other is g^c (for a random c). The task is to tell which of (z, z') is $g^{a \cdot b}$.

The Bellare-Micali Construction

- Initialization: The sender chooses a random C in Z_q .
- Protocol:
 - The receiver picks a random $k \in [1, q]$, sets public keys $PK_j = g^k$, and $PK_{1-j} = C/PK_j$. It sends PK_0 to the sender.
 - The sender computes $PK_1 = C/PK_0$. Chooses random r_0, r_1 .
 - Generates El Gamal encryptions: $E_0 = (g^{r_0}, H((PK_0)^{r_0}) \oplus x_0)$, $E_1 = (g^{r_1}, H((PK_1)^{r_1}) \oplus x_1)$ and sends them to the receiver.
 - The receiver computes $H((PK_j)^{r_j})$ and decrypts E_j .
- Security:
 - Sender cannot learn anything about j (unconditionally).
 - It's obvious that the receiver cannot compute the discrete logs of both PK_0 and PK_1 .

The Bellare-Micali Construction

- A semi-honest receiver
 - Picks a random $k \in [1, q]$, sets public keys $PK_j = g^k$, and $PK_{1-j} = C/PK_j$. (I.e., picks PK_{1-j} at random.)
 - $E_j = (g^{r_j}, H((PK_j)^{r_j}) \oplus x_j)$
 - If the DDH assumption holds, the receiver cannot distinguish $(PK_j)^{r_j}$ from random.
 - A malicious receiver:
 - Cannot know discrete logs of both PK_0 and PK_1 .
 - But, might know some *partial* information about the discrete log of both PK_0 and PK_1 . (We don't know how it can do this, but this information might reveal something about the plaintexts.)
 - Security can be proven assuming that $H()$ is a random oracle.

OT without random oracles

- Security is based on the DDH assumption alone.
- Z_p^* , q , and sender's and receiver's inputs are as before.
- Receiver
 - chooses random $a, b, c_{1-j} \in [1, q]$, and defines $c_j = ab \pmod{q}$.
 - It sends to the sender $(g^a, g^b, g^{c_0}, g^{c_1})$.
- The sender
 - Certifies that $g^{c_0} \neq g^{c_1}$. Chooses random $s_0, r_0, s_1, r_1 \in [1, q]$.
 - Defines $w_0 = (g^a)^{s_0} g^{r_0}$. Encrypts x_0 with the key $(g^{c_0})^{s_0} (g^b)^{r_0}$.
 - Defines $w_1 = (g^a)^{s_1} g^{r_1}$. Encrypts x_1 with the key $(g^{c_1})^{s_1} (g^b)^{r_1}$.
 - Sends w_0, w_1 and the encryptions to receiver.
- Receiver computes $(w_j)^b$ and decrypts x_j .

Security

- Receiver's security is based on the DDH assumption: sender cannot decide if g^{c0} or g^{c1} are equal to g^{ab} .
- Sender security is unconditional:
 - Suppose that $j=0$. Need to show that x_1 is secure.
 - Denote $s=s_1$, $r=r_1$.
 - Receiver obtains $w_1=(g^a)^s g^r = g^{as+r}$, and encryption with the key $k_1=(g^c)^s (g^b)^r = g^{cs+br}$. It's also known that $c \neq ab$.
 - We will show (on the board) that regardless of the choice of a, b, c , the pair (w_1, k_1) is uniformly distributed.
 - Therefore w_1 leaks nothing about the key k_1 .

1-out-of-N OT

- A generalization of 1-out-of-2 OT:
 - Sender has N inputs, x_0, \dots, x_N .
 - Receiver has an input $j \in \{1, 2, \dots, N\}$.
- Output:
 - Receiver learns x_j and nothing else.
 - Sender learns nothing about j .
- We would like to construct 1-out-of-N OT, or reductions from 1-out-of-N OT to 1-out-of-2 OT.
 - It was shown that any such reduction which provides unconditional security requires at least $N-1$ OTs.
 - Since OT has a high computational overhead, we would like to do better than that.

A recursive protocol for 1-out-of-N OT

- The reduction uses a pseudo-random function $F_k()$.
 - It holds that if k is chosen at random and kept secret, no adversary can distinguish between $(x, F_k(x))$ and a random value, for every x .
- The protocol:
 - Assume $N=m \times m$. Denote inputs as $\{x_{i,j}\}_{1 \leq i,j \leq m}$.
 - The sender prepares $2m$ keys, $R_1, \dots, R_m, C_1, \dots, C_m$. It computes encryptions $Y_{i,j} = X_{i,j} \oplus F_{R_i}(j) \oplus F_{C_j}(i)$.
 - The receiver runs two 1-out-of- m OT protocols to learn R_i and C_j . Corresponding to the item that it wants to learn.
 - The sender sends all $Y_{i,j}$ to the receiver, which decrypts the item it is interested in.
- We can reduce 1-out-of- m OT to 1-out-of- \sqrt{m} OT, etc.