Advanced Topics in Cryptography

Lecture 1

Benny Pinkas

Administrative Details

- Grade
  - Based on
    - homeworks (might include a final take home exam)
    - Participation in class
- Office hours: Sundays, 15-16.
- Email: benny@cs.haifa.ac.il
- Web page: http://www.pinkas.net/courses/atc/2008/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
  - 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* AND *b* without revealing any other information about their inputs.

- If P1’s input is *a=0,* and he learns that *(a* 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?
  - dating

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 b=1 then the result obtained in the OT protocol is equal to *a*, but it is also equal to $a\cdot b$ which is the legitimate output of P2.
  - P1 need not worry if P2’s input is 1.

The Dating Problem

Is a=b?

Leak no other information!

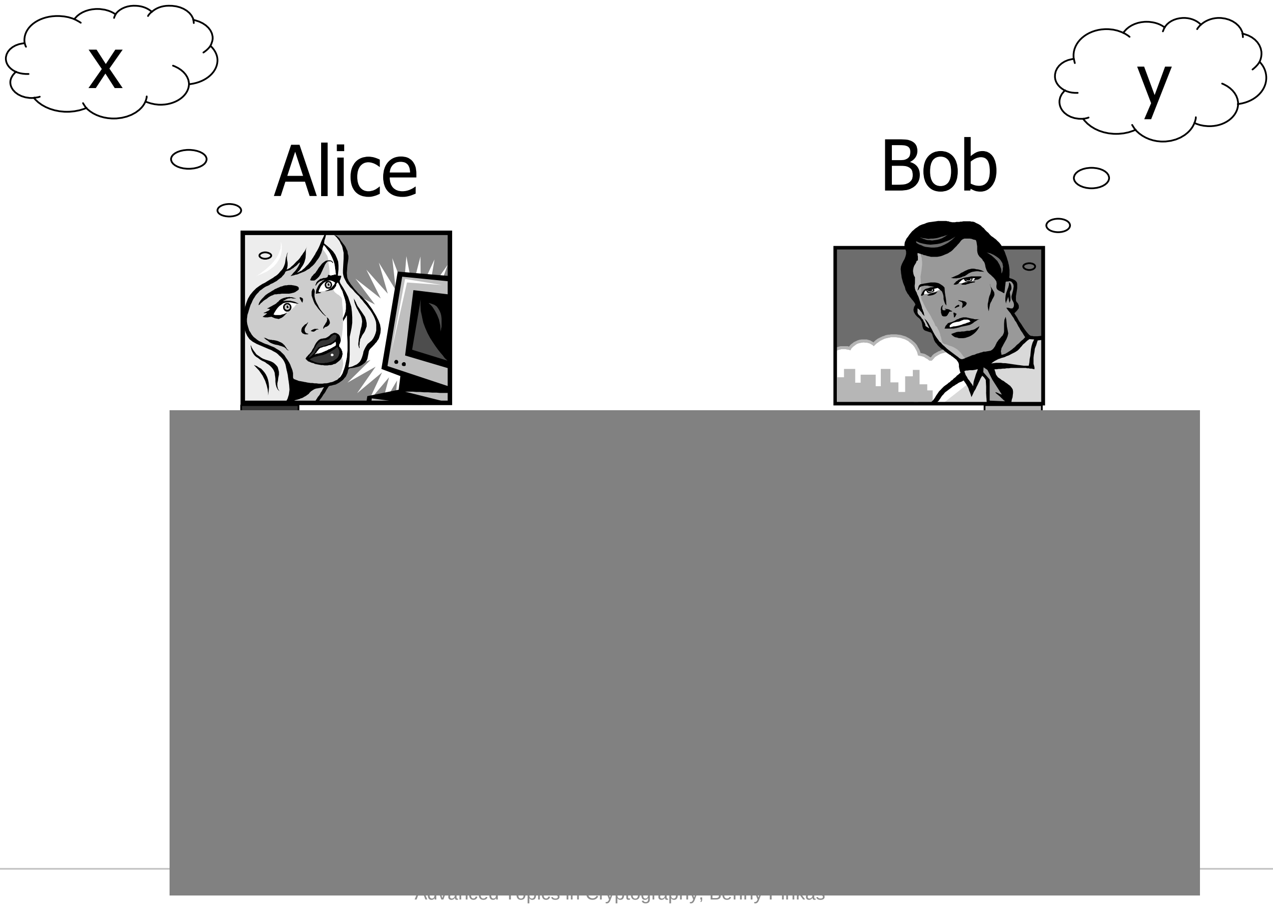
The Millionaires Problem

x
y
Alice
Bob

Whose value is greater?

Leak no other information!

Ideal Solution for the Secure Computation Problem



Secure Function Evaluation (Informal) Definition

A protocol is secure if it emulates the ideal solution (the participants do not learn any information that they do not learn in the ideal model)

OR

For any adversary there is a comparable one working in the Ideal Model with ***similar*** output

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=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: comparison to the ideal model

- Now we must state what it means to learn nothing but the output of the protocol.
- Best definition: comparison to the ideal model
  - Security according to this definition is easy to achieve against semi-honest adversaries but hard to achieve against malicious adversaries.
  - Ideal implementation:
    - There is a third trusted party (TTP). It receives the inputs from the parties, computes the output, and sends it to the parties.
    - For OT: the TTP receives x_0,x_1 from the sender, and j from the receiver. It sends x_j to the receiver.
- Security – comparison with the ideal model
  - Given a player’s interaction with the TTP, it is possible to simulate the interaction it has in the protocol.

Security of Computing “AND” Privately using OT

- Privacy: (against semi-honest adversaries)
  - 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 sender (P1) in the OT protocol. Therefore P2’s output in the OT protocol is a function of the “AND” result.
    - Our input: a AND b
    - We set $x_0=0$, $x_1=a$ 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?

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.
- – In a certain protocol execution with a=0, the protocol has transcript T and output “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 therefore 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.

Security Definitions for OT

- It appeared to be quite hard to design an OT protocol that is secure against malicious adversaries in the sense of comparison to the ideal model

- Therefore looser security definitions were used
  - Defining what is means to protect the receiver’s security is easy, since the sender receives no output in the ideal model and should therefore learn nothing about the receiver’s input.
  - As for sender’s security, this is more tricky.
  - See next slide.

Security Definitions for OT

- Receiver’s security – indistinguishability
  - For any values of the sender’s inputs 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.

Security Definitions for OT

- 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.
- This definition does 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

The Even-Goldreich-Lempel 1-out-of-2 OT construction (providing security only against semi-honest adversaries)

- 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
(providing security against malicious adversaries)

- 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 to the problem is 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 to the problem is 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: (slightly modified)
  - 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 a random r.
  - Generates El Gamal encryptions: $E_0 = (g^r, H((PK_0)^r) \oplus x_0)$, $E_1 = (g^r, H((PK_1)^r) \oplus x_1)$ and sends them to the receiver.
  - The receiver computes $H((PK_j)^r)$ and decrypts E_j.
- Security:
  - Sender cannot learn anything about j (unconditionally).
  - It is obvious that the receiver cannot compute the discrete logs of both PK_0 and PK_1.

Security of the Bellare-Micali Construction

- The receiver cannot compute the discrete logs of both PK_0 and PK_1.

- The Diffie-Hellman assumption implies that it cannot compute both $(PK_0)^r$ and $(PK_1)^r$, since
  - Computing both $(PK_0)^r$ and $(PK_1)^r$, implies that the receiver can also compute C^r.
  - Namely, the receiver who knows g,C,g^r (for random C and r) can compute C^r. This is a contradiction to the Diffie-Hellman assumption.
- There is therefore an index i such that the receiver does not know $(PK_i)^r$
  - If we assume that $H()$ is a random function (a random oracle) then the receiver cannot distinguish $H(PK_i)^r$ from a random string.

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 $c_j = ab \ (mod\ q)$.
  - It sends to the sender $(g^a, g^b, g^{c0}, g^{c1})$.
- The sender
  - Certifies that $g^{c0} \neq g^{c1}$. Chooses random $s_0, r_0, s_1, r_1 \in [1,q]$.
  - Defines $w_0 = (g^a)^{s0} g^{r0}$. Encrypts x_0 with the key $(g^{c0})^{s0}(g^b)^{r0}$.
  - Defines $w_1 = (g^a)^{s1} g^{r1}$. Encrypts x_1 with the key $(g^{c1})^{s1}(g^b)^{r1}$.
  - 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, $x_0,\ldots x_{N-1}$.
  - Receiver has an input $j \in \{0,1,\ldots,N-1\}$.
- Output:
  - Receiver learns x_j and nothing else.
  - Sender learns nothing about j.
- We would like to construct 1-out-of-N OT, or a reduction that uses 1-out-of-2 OT for constructing 1-out-of-N OT.
  - It was shown that any such reduction which provides unconditional security requires at least N-1 OTs.

Implementing 1-out-of-N OT using N invocations of 1-out-of-2 OT

- The protocol:
  - Let E(k,x) denote the symmetric encryption of x using the key k (say, using DES or AES).
  - Sender chooses N keys, $k_0, \dots k_{N-1}$.
  - $\forall i$ it computes $c_i = E(k_{N-1}, \dots E(k_{i+1}, E(k_{i-1}, \dots E(k_0, x_i) \dots)$
  - The parties run N 1-out-of-2 OTs:
    - In the j'th invocation the sender's input is the pair $\langle k_j , c_j \rangle$
  - A receiver that is interested in learning x_i chooses to learn the values $k_0, \dots, k_{i-1}, c_i, k_{i+1}, \dots, k_{N-1}$.

- Security?
- Since OT has a high computational overhead, we would like to do better than that. (And there are more efficient protocols.)

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 $\{ x_{i,j} \}_{1\le i,j \le m}$.
  - The sender prepares 2m keys, $R_1,\ldots,R_m,C_1,\ldots,C_m$. 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.