Secure Computation

Lecture 1

Benny Pinkas

Administrative Details

- Grade
  - Based on
    - Homeworks 70%
    - A final take home project/large homework/exam 30%
    - Participation in class
- Email: benny@pinkas.net
- Web page: http://www.pinkas.net/courses/sc/index.html
- Goal: Learn how to perform a distributed computation while not revealing the inputs

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.

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
  - Efficient Secure Two-Party Protocols", Hazay and Lindell, 2010. Ch. 7.

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.

The Input Equality Problem

Is a=b?

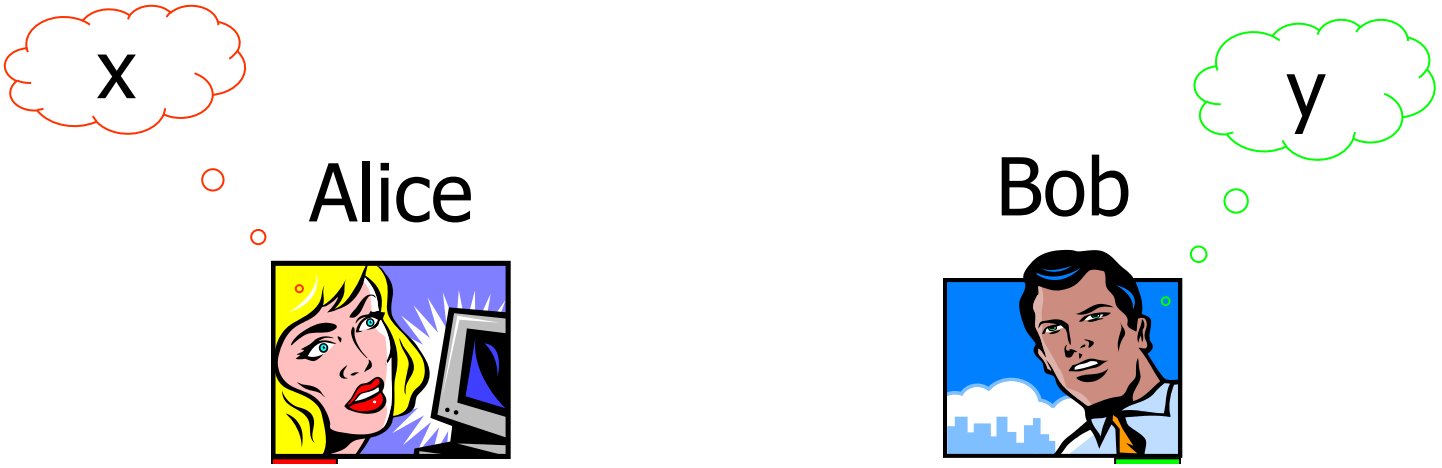
Leak no other information!

The Millionaires Problem

Whose value is greater?

Leak no other information!

Ideal Solution for the Secure Computation Problem



What properties would we want of secure computation?

- Privacy: No party should learn anything more than its prescribed output (and anything derived from it).
- Correctness: The output of all parties is correct.
- Independence of inputs: Corrupted parties must choose their inputs independently of the honest parties' inputs. (Think about auctions.)
- Fairness: Corrupted parties receive their outputs if and only if the honest parties also receive their outputs. (Think about signing contracts.)

Secure Computation (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 $z=Enc(y)$.

- 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

- Must state what it means to learn nothing but the output of the protocol.
  - Intuitively, a protocol is secure if whatever can be computed by a party participating in the protocol can be computed based on its input and output only.

- Comparison to the ideal model
  - Ideal implementation:
    - There is a trusted third 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.
    - For AND: TTP receives a,b from P1,P2, and returns a *AND* b.
  - 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 a corrupt P2: (against semi-honest adversaries)
  - Show here that P2 does not learn more than the "AND" result.
  - Must show 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 can simulate 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 alone.
    - Our input: a AND b, and P2's input 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?

Security of Computing "AND" Privately using OT

- Privacy against a corrupt P1: (against semi-honest adversaries)
  - We show that P1 does not learn more than the "AND" result.
  - Proof: Recall that P2 is the receiver in the OT. After receiving the output of the OT it sends it to P1.
  - We receive the output of the "AND" function and construct the input of the receiver in the OT protocol. Therefore P1's output in the OT protocol is a function of the "AND" result.
    - Our input: a AND b, and P1's input a.
    - We set the receiver's input to be j=a AND b.
    - P1's input to the OT is, as in the original protocol, (0,a).
    - If (a AND b) is 0 then the output of the protocol is 0; if (a AND b) is 1 the output of the protocol is a (and in this case it holds that a=1). Therefore the output is as in the original protocol.

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 OT with unconditional security

- Suppose that there is an AND protocol (between P_1 and P_2, with inputs a and b) with unconditional security.
  - Such a protocol could be constructed from an OT which has unconditional security.
- Let T be a transcript of all messages sent in the protocol.
- The parties use random inputs R_1 and R_2.
  - Given these inputs the transcript T is a deterministic function.

Impossibility of achieving OT with unconditional security

- In a certain execution with P_1's input a=0, the protocol has transcript T and output "0".
  - If b=0, then P_2 must not learn P_1's input.
  - Therefore $\exists$ an R'_1 s.t. if P_1 has inputs a=1 and R'_1, the protocol would have produced the same transcript T.
  - If b=1, then output is 0. Therefore there is no R''_1 s.t. the protocol has transcript T for a P_1 input of a=1 (this is because the output of the protocol in this case is "1").

- P_1 can therefore
  - search over all possible values for R_1 and check if running them with input a=1 results in transcript T.
  - If there is such an R_1 then it concludes that b=0.