Data Security and Privacy

Secure Function Evaluation

Outline and Readings

- Outline:
  - 1-out-2 Oblivious Transfer
  - Private Information Retrieval
  - Yao's Scrambled Circuits for 2-party SFE
  - Secret Sharing
  - n-party SFE

- Readings:

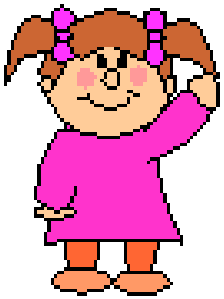
Oblivious Transfer

- 1 out of 2 OT
  - Alice has two messages x_0 and x_1
  - At the end of the protocol
    - Bob gets exactly one of x_0 and x_1
    - Alice does not know which one Bob gets

- 1 out of n OT
  - Alice has n messages
  - Bob gets exactly one message, Alice does not know which one Bob gets
  - 1 out of 2 OT implies 1 out of n OT

Bellare-Micali 1-out-2-OT protocol

g: generator of G_q, a group of order q



x_0, x_1

$b \in \{0,1\}$

$c \leftarrow_R G_q$

c →

$k \leftarrow_R Z_q$

$z_b = g^k$

$z_{1-b} = c/g^k$

← z_0, z_1

$C_0 = [g^{r0}, H(z_0^{r0}) \oplus x_0]$,

$C_1 = [g^{r1}, H(z_1^{r1}) \oplus x_1]$ →

decrypts $C_b = [v_1, v_2]$ by computing $H(v_1^k) \oplus v_2$

B gets only one of $\{x_0, x_1\}$, w/o A knowing which one it is.

OT versus Private Information Retrieval (PIR)

- A Private Information Retrieval (PIR) protocol enables client B to retrieve one entry from a database maintained by server A without A knowing which entry it is
  - Achieves 1-out-of-*n* Oblivious Transfer with communication cost sub-linear in *n*
- May relaxes the requirement that B retrieves only one entry in the OT requirement
  - Naïve protocol is for A to send everything to B.
  - Want to design protocol with sub-linear communication cost.
  - Sub-linear computation cost for the server (A) impossible
    - A must scans through the whole database, otherwise A learns sth about what has been accessed.
    - With multiple non-colluding servers, sub-linear computation is possible

Secure Function Evaluation

- Also known as Secure Multiparty Computation
- 2-party SFE: Alice has x, Bob has y, and they want to compute two functions $f_A(x,y)$, $f_B(x,y)$. At the end of the protocol
  - Alice learns $f_A(x,y)$ and nothing else
  - Bob learns $f_B(x,y)$ and nothing else
- n-party SFE: n parties each have a private input, and they join compute functions

Adversary Models

- There are two major adversary models for secure computation: Semi-honest model and fully malicious model.
  - Semi-honest model: all parties follow the protocol; but dishonest parties may be curious to violate others' privacy.
  - Fully malicious model: dishonest parties can deviate from the protocol and behave arbitrarily.
    - Clearly, fully malicious model is harder to deal with.

Security in Semi-Honest Model

- A 2-party protocol between A and B (for computing a **deterministic function f()**) is secure in the semi-honest model if there exists an efficient algorithm MA (resp., MB) such that
  - the view of A (resp., B) is **computationally indistinguishable** from MA(x1,f1(x1,x2)) (resp., MB(x2,f2(x1,x2)).
- We can have a similar (but more complex) definition for multiple parties.

Security in Malicious Model (1)

- In the malicious model, security is much more complex to define.
- For example, there are unavoidable attacks:
  - What if a malicious party replaces his private input at the very beginning?
  - What if a malicious party aborts in the middle of execution?
  - What if a malicious party aborts at the very beginning?

Security in Malicious Model (2)

- To deal with these complications, we use an approach of ideal world vs. real world.
  - Consider an ideal world in which all parties (including the malicious ones) give their private inputs to a trusted authority.
  - After receiving all private inputs, the authority computes the output and sends it to all parties.
  - Clearly, those unavoidable attacks also exist in this ideal world.

Security in Malicious Model (3)

- We require that, for any adversary in the real world, there is an “equivalent” adversary in the ideal world, such that
  - The outputs in the real world are computationally indistinguishable from those in the ideal world.
- In this way, we capture the idea that
  - All “avoidable” attacks are prevented.
  - “Unavoidable” attacks are allowed.

Yao's Theorem

- The first completeness theorem for secure computation.
- It states that for ANY efficiently computable function, there is a secure two-party protocol in the semi-honest model.
  - Therefore, in theory there is no need to design protocols for specific functions.
  - Surprising!

Yao's Scrambled Circuit Protocol for 2-party SFE

- For simplicity, assume that Alice has x, Bob has y, Alice learns f(x,y), and Bob learns nothing
  - represent f(x,y) using a boolean circuit
  - Alice encrypts the circuit and sends it to Bob
    - in the circuit each wire is associated with two random values
  - Alice sends the values corresponding to her input bits
  - Bob uses OT to obtain values for his bits
  - Bob evaluates the circuits and send the result to Alice

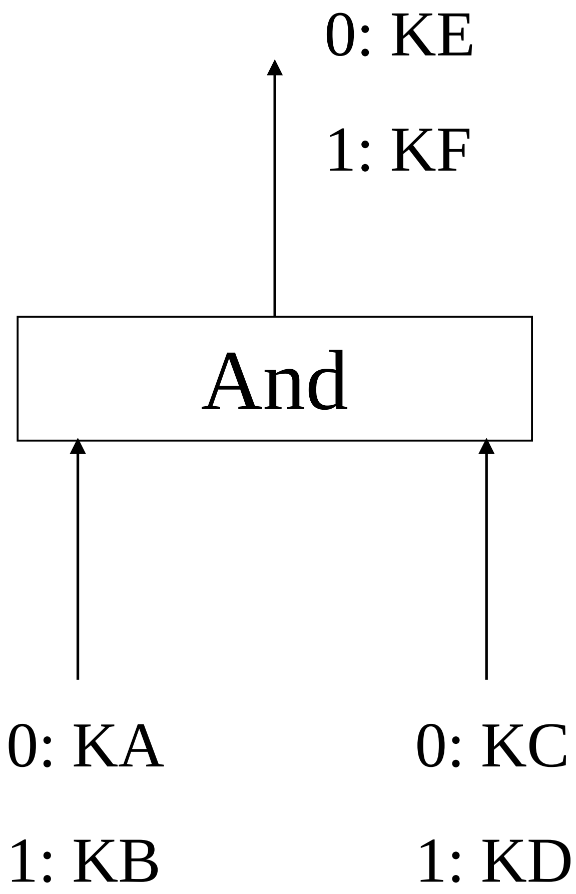
Circuit Computation

- The design of Yao's protocol is based on circuit computation.
  - Any (efficiently) computable function can be represented as a family of (polynomial-size) boolean circuits.
  - Such a circuit consists of and, or, and not gates.

Garbled Circuit

- We can represent Alice's circuit with a garbled circuit so that evaluating it does not leak information about intermediate results.
  - For each edge in the circuit, we use two random keys to represent 0 and 1 respectively.
  - We represent each gate with 4 ciphertexts, for input (0,0), (0,1), (1,0), (1,1), respectively.
    - These ciphertexts should be permuted randomly.
  - The ciphertext for input (a,b) is the key representing the output Gate(a,b) encrypted by the keys representing a and b.

Example of a Gate



- This gate is represented by:

  (a random permutation of)

  $E_{KA}(E_{KC}(KE))$;
  $E_{KB}(E_{KC}(KE))$;
  $E_{KA}(E_{KD}(KE))$;
  $E_{KB}(E_{KD}(KF))$.

Evaluation of Garbed Circuit

- Given the keys representing the inputs of a gate, we can easily obtain the key representing the output of the gate.
  - Only need to decrypt the corresponding entry.
  - But we do not know which entry it is? We can decrypt all entries. Suppose each cleartext contains some redundancy (like a hash value). Then only decryption of the right entry can yield such redundancy.

Translating Input?

- So, we know that, given the keys representing Bob's private input, we can evaluate the garbled circuit.
  - Alice sends the garbled circuit, and the keys corresponding to her input.
  - Then Bob can evaluate the garbled circuit if he knows how to translate his input to the keys.
- But Alice can't give the translation table to Bob.
  - Otherwise, Bob can learn information during evaluation.

Jump Start with Oblivious Transfer

- A solution to this problem is 1-out-of-2 OT for each input bit.
  - Alice sends the keys representing 0 and 1;
  - Bob chooses to receive the key representing his input at this bit.
  - Clearly, Bob can't evaluate the circuit at any other input.

Finishing the Evaluation

- At the end of evaluation, Bob gets the keys representing the output bits of circuit.
  - Alice sends Bob a table of the keys for each output bit.
  - Bob translates the keys back to the output bits.

Secret Sharing

- t-out-of-n secret sharing
  - divides a secret s into n pieces so that any t pieces together can recover n
- How to do n-out-of-n secret sharing?
- Shamir's secret sharing scheme
  - secret $s \in Z_p$
  - pick a random degree t-1 polynomial $f \in F_p[x]$ s.t. $f(0)=s$
  - user i gets $s_i=f(i)$
  - t users can interpolate f and find out b
  - t-1 shares reveal no information about s

Proactive Secret Sharing

- Suppose that s is shared in t-out-of-n
- User i has $s_i=f(i)$
- Proactive updates:
  - user 1 picks random degree t-1 polynomial s.t. g(t)=0
  - user 1 sends $y_j=g(j)$ to user j
  - user j does $s_j^{new}=s_j^{old}+y_j$

BGW n-party SFE

- Use algorithmic circuits where operations are + and $\times$
  - All computable functions can be represented as an algorithmic circuit
- Each private input is shared among all participants
- Do computation with the shared value
  - e.g., given x and y both are shared by n parties, compute the shares of x+y and $x \times y$
- Secure when the majority of the parties are honest

From Semi-Honest to Malicious

- Based on general-purpose protocols in the semi-honest model, we can construct general-purpose protocols in the malicious model.
  - The main tools are bit commitment, (verifiable) secret sharing, and zero-knowledge proofs.
  - In fact, “compilers” are available to automatically translating protocols.

Coming Attractions …

- Topics
  - Identity based encryption & quantum cryptography