Advanced Topic: Cryptographic Protocols

Sheng Zhong

Outline

- Bit Commitment
- Secret Sharing
- Oblivious Transfer
- Secure Computation

Bit Commitment (1)

- Suppose Alice and Bob want to flip a coin to decide something.
  - However, they are not physically in the same place.
  - How can they flip a coin over the phone?
  - If Alice flips the coin, she might want to manipulate the result so that it is to her favor.
  - If Bob flips the coin, he might do the same thing.

Bit Commitment (2)

- One possible solution is:
  - Alice flips a coin and commits to it.
  - Bob flips another coin and tells Alice his result.
  - Alice reveals her own result and the final result= Alice’s result xor Bob’s result.
- But how can Alice commit to a bit?

Bit Commitment Scheme

- A bit commitment scheme allows Alice to compute a commitment of a bit, such that:
  - Alice can reveal the value of this bit later.
  - Alice cannot cheat (i. e., give a false value) when revealing the value of this bit.
  - Bob cannot compute the value of this bit from the commitment.
  - A cryptographic hash value will work as a commitment.
  - But we now seek for a better solution, which allows algebraic operations and randomness on commitments.

Example: Bit Commitment based on Discrete Logarithm

- An example of bit commitment scheme:
  - Let p be a large prime.
  - Let g be a generator of Zp*.
  - Commitment to 0: g^x, where x is a uniform random number in [0, (p-1)/2).
  - Commitment to 1: g^x, where x is a uniform random number in [(p-1)/2, p-1).
  - The scheme is secure under the assumption that discrete log is hard.

General Commitment

- More generally, we can commit to a bit string or an integer rather than to a single bit.
- Example Scheme (by Chaum):
  – Let g and h be two generators mod large prime p, picked independently.
  – Commitment to x: $g^x h^r$, where r is a random number.

Secret Sharing

- Suppose a company has a very important secret. Who should know this secret?
  - If only the CEO knows it, then what if something unexpected happened to him?
  - If a good number number of people (e.g., all directors) know it, then what if one of them were corrupted?
  - A cryptographic solution to this problem is secret sharing.

Secret Sharing Scheme

- A secret sharing scheme allows a secret s to be shared among n parties with a threshold t, such that:
  - Any group of t parties can easily recover s.
  - Any group of <t corrupted parties cannot figure out s.
- The above scenario often needs to be established by a trusted third party or using a special method.

Shamir Secret Sharing

- The first secret sharing scheme was proposed by Adi Shamir.
  - Choose a random degree-(t-1) polynomial f() with the constant term=s.
  - Choose n points $x_1, \ldots, x_n$ ($\neq 0$).
  - The ith party has share: $f(x_i)$.
  - To recover s only needs to interpolate the polynomial using t points.
  - <t points have no information about s.

Verifiable Secret Sharing

- How can I know whether a share is correct or not?
  - Note that the correctness of a share can be verified using t other shares.
  - However, we can’t ask other parties to reveal t shares.
- So each share should have a commitment which is public.
  - The correctness of shares can be verified using commitments.
  - This is called Verifiable Secret Sharing (VSS).

Oblivious Transfer

- A simple cryptographic primitive first studied by Rabin.
  – Kilian showed that you can essentially base ANY cryptographic protocol on this primitive.
- Suppose Alice sends a message to Bob.
  – We want that Bob receives the message with probability ½.
  – We also want that Alice does not know whether Bob receives it or not.

Rabin's OT Protocol (1)

- Alice chooses an RSA modulus N=pq and the a pair of encryption/decryption exponents (e,d).
- Alice encrypts message m using RSA under key (N, e).
- Alice sends the ciphertext, N, e to Bob.
- Bob chooses a in Z_N* and computes $b=a^2$ mod N.
- Bob sends the b to Alice.

Rabin’s OT Protocol (2)

- Alice computes the four square roots of b mod N.
  - Recall this is done by computing the square roots mod p and mod q respectively, and then using the Chinese Remainder Theorem.
- Alice chooses one of the square roots uniformly at random and sends it back to Bob.
- Bob checks whether he has received a or –a.
  - If yes, he can’t get m.
  - If no, he can factor N and compute m.

Security Analysis

- Bob indeed gets m with probability ½.
  - Because the probability of picking a or –a from the four square roots is ½.
  - And because if a or –a is sent to Bob, then Bob gets no help in factoring N.
- Alice has no way to learn whether Bob gets m or not.
  - Because she has no idea which of the four roots is a.

1-out-2 OT

- There are many variants of OT; 1-out-of-2 OT is a popular one.
  - Alice has two messages m_0 and m_1.
  - Bob has a bit b (i.e., chooses to receive m_b).
  - Alice should not learn b.
  - Bob should not learn m_{1-b}.

Bellare-Micali Protocol (1)

- Let G be a cyclic group in which discrete log is hard; let g be a generator.
- Alice (or a public procedure) chooses y in G.
- Bob chooses x_b and computes $y_b=g^{x_b}$.
- Bob also computes $y_{1-b}=y/y_b$.

Bellare-Micali Protocol (2)

- Bob sends y_0, y_1 to Alice.
- Alice checks $y_0 y_1 = y$ and encrypts m_0, m_1 using ElGamal public parameters y_0, y_1, respectively.
- Bob decrypts the encryption of m_b using x_b.

Security Analysis

- Bob can't learn m_{1-b} because he does not know the discrete log of y_{1-b}.
  – Guaranteed by the security of ElGamal cryptosystem.
- Alice can't learn b because everything she observes is independent of b.

1-out-of-2 OT implies OT

- Suppose we have a 1-out-of-2 OT protocol. Then we can construct an OT protocol.
  - Alice randomly permutes (m, trash).
  - Alice runs 1-out-of-2 OT with Bob with the above permuted pair.
  - Regardless of Bob’s choice in 1-out-of-2 OT, he always receives m with probability ½.

1-out-of-n OT

- An extension of 1-out-of-2 OT.
  - Alice has n messages.
  - Bob has an input in [0,n-1] (i.e., chooses to receive one of the messages).
  - Alice should not learn Bob’s choice.
  - Bob should not learn the other n-1 messages.

1-out-of-2 OT implies 1-out-of-n OT (1)

- Suppose we have a 1-out-of-2 OT protocol. Then we can construct a 1-out-of-n OT protocol.
- Let's temporarily assume $n=2^m$.
  - Alice chooses 2m random numbers k_1, k'_1 …, k_m, k'_m.
  - For each message m_i, for each j: if the jth bit of i is 0, then the message is xor'd by k_i; otherwise the message is xor'd by k'_i.
  - For example: i=1011, then m_i is xor'd by k_1 xor k'_2 xor k_3 xor k_4.

1-out-of-2 OT implies 1-out-of-n OT (2)

- For each j, Alice and Bob run a 1-out-of-2 OT protocol such that Bob learns one of the two random numbers k_j and k'_j.
- The random numbers Bob learns can only help him learn one message.
- Alice clearly can't learn Bob's choice.

- But what if n is not a power of 2?

1-out-of-2 OT implies 1-out-of-n OT (3)

- When n is not a power of 2, let n' be the smallest power of 2 such that n'>n.
  - Alice runs a 1-out-of-n' OT protocol with Bob using the n messages and n'-n pieces of trash.
  - Alice tells Bob where the n'-n pieces of trash are, so that Bob would not choose to receive any of them.
  - Clearly, Bob will receive a message of his choice; Alice won't learn Bob's choice; Bob won't learn other messages.

Secure Computation

- Secure 2-party/multi-party computation: general-purpose cryptographic protocol.
  - Suppose there are n parties.
  - A common public input: function f().
    - f()=(f1(),f2())
  - Each party has a private input x_i.
  - Can we construct a protocol for securely computing $f(x_1, \ldots, x_n)$?
    - A should only learn $f1(x_1, \ldots, x_n)$;
    - B should only learn $f2(x_1, \ldots, x_n)$.

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!

The Setting

- Yao's theorem applies to the following setting of **Secure Function Evaluation**:
  - Alice has a function f(), which is efficiently computable.
  - Bob has an input x.
  - We need a way to evaluate f(x) such that
    - Alice learns nothing;
    - Bob learns only f(x).

Secure Function Evaluation vs. Secure Two-Party Computation

- We can view Secure Function Evaluation of f() as a special case of Secure Two-Party Computation.
  - Because f() is also an input, after all.
- We can also build Secure Two-Party Computation of F() based on Secure Function Evaluation.
  - Just define f(y)=F2(x,y) and evaluate f().
  - Then define f'(x)=F1(x,y) and evaluate f'().
- So Secure Function Evaluation is essentially equivalent to Secure Two-Party Computaiton.

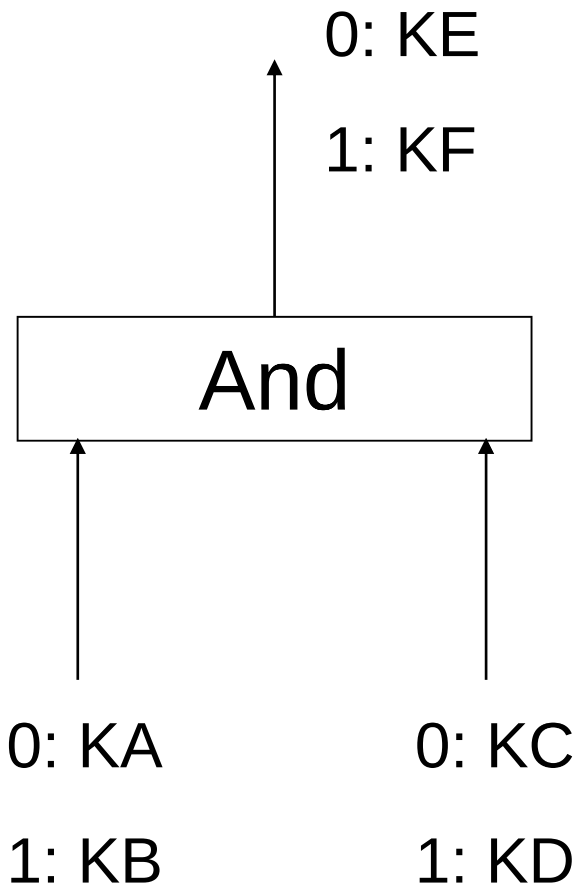
Circuit Computation

- The design of Yao’s protocol is based on circuit computation.
  - Recall any (efficiently) computable function can be represented as a family of (polynomial-size) boolean circuits.
  - Recall such a circuit consists of and, or, and not gates.
  - It is enough if we can evaluate Alice’s private circuit at Bob’s private input.

Garbled Circuit

- We can represent Alice's circuit with a garbled circuit that does not reveal any knowledge about the circuit.
  - 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.
  – Suppose Alice also sends the garbled circuit to Bob. 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 evaluate the circuit at ANY input.

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.
- For privacy, we need to be careful:
  - The topology of the circuit should be the same for all circuits of a particular input size.
  - Then privacy is guaranteed.

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.