Advanced Topics in Cryptography

Lecture 7

Benny Pinkas

Goal

- A 1-out-of-2 OT protocol fully secure against malicious adversaries
- Namely,
  - Such that for *every adversary **A*** attacking the real protocol, *there exists an adversary **S*** in the ideal model such that the output distributions (of all parties) are computationally indistinguishable

OT ensuring privacy alone against malicious adversaries [NP]

- We learned this protocol in a previous lecture

- Security is based on the DDH assumption alone.
  - Security is proven according to a definition that ensures only privacy, rather than full security.
  - Receiver's privacy – 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.
  - Definition of sender's security:
    - $\forall$ receiver A' $\exists$ algorithm A" in the ideal model s.t. $\forall$ x_0,x_1 the outputs of A' and A" are indistinguishable.
    - Namely, A' learns nothing more than A".

Recap: OT secure against malicious adversaries, without random oracles [NP]

- Z_p^*, q 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 $(g^a, g^b, g^{c0}, g^{c1})$.
- 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 $(w_j)^b$ which is the key with which x_j was encrypted. It uses it to and decrypt x_j.

Properties

- Correctness
  - Suppose j=0. R sends (g^a, g^b, g^{ab}, g^c).
  - S defines $w_0=(g^a)^{u0}g^{v0}$.
  - S encrypts x_0 with $k_0=(g^{ab})^{u0}(g^b)^{v0}$.
    - Note that encryption key is equal to $(w_0)^b$.
  - R computes $k_0=(w_0)^b$ and uses it for decryption.

Privacy

- Receiver's security
  - Show that sender's view is indistinguishable regardless of receiver's input.
    - Sender receives either (g^a, g^b, g^{ab}, g^c) or (g^a, g^b, g^c, g^{ab}).
    - Distinguishing them implies breaking the DDH assumption
- The security of the sender is unconditional.
  - Suppose that j=0. Regarding x_1, the server sends $w_1=(g^a)^{s1}g^{r1}$. x_1 is then encrypted with the key $k_1=(g^{c1})^{s1}(g^b)^{r1}$.
  - The values s_1, r_1 were chosen at random, and $ab \neq c_1$.
  - We showed that (w_1, k_1) are uniformly distributed, and can be easily simulated.

Why doesn't this protocol provide full security?

- We would like to describe for each party a simulator that can interact with that party in a way that is indistinguishable from the protocol.
  - The simulator can access a trusted party, that when given the party's input provides the right output
- With regards to a receiver
  - The simulator must extract the receiver's input and send it to the trusted party
- With regards to a sender
  - The simulator must extract both of the sender's inputs and send them to the trusted party

Background: Zero-knowledge proofs of knowledge

- Learned about them earlier in the course

Approaches for a solution

- Handling a corrupt **receiver**.
- The receiver sends $(g^a, g^b, g^{c0}, g^{c1})$
- Ask the receiver to send a zk proof of knowledge for a. Namely, prove that it knows the discrete log of g^a to the base g.
- Then the simulator can extract a from this proof, and use it to find out which one of g^{c0}, g^{c1} is equal to g^{ab}.
  - This reveals the receiver's input, that can then be sent to the trusted party.

Approaches for a solution

- Handling a corrupt **sender**.
- The receiver sends $(g^a, g^b, g^{c0}, g^{c1})$
- Only one of g^{c0} or g^{c1} is equal to g^{ab}.
- For the other option, the message that the sender encrypts and the receiver receives is unconditionally secure, and therefore the sender's input cannot be extracted from it…
- Must somehow solve this issue

OT protocol with full security

- Inputs: Sender has inputs x_0, x_1. Receiver has input b.

- Auxiliary input: A group G of prime order q (so that each element is a generator), and a generator g.

OT protocol with full security

- R (with input j) chooses random a_0,a_1,r in [1,q]. Computes
  - $h_0=g^{a0}$, $h_1=g^{a1}$, $a=g^r$, and $b_0=(h_0)^r g^j$, $b_1=(h_1)^r g^j$ and sends them.
- Let $h=h_0/h_1$, and $b=b_0/b_1$. The receiver proves that (g,h,a,b) is a Diffie-Hellman tuple with relation to (G,q).
- (Indeed $h=g^{a0-a1}$, $b=g^{(a0-a1)r}$, and $a=g^r$, so this is DDH.)
- Note that this means that
  - when j=0, (g,h_0,a,b_0) is a DDH tuple and $(g,h_1,a,b_1/g)$ is **not** a DDH tuple
  - and when j=1, (g,h_0,a,b_0) is **not** a DDH tuple and $(g,h_1,a,b_1/g)$ is a DDH tuple.
  - If the proof is verified, then it cannot be that both (g,h_0,a,b_0) and $(g,h_1,a,b_1/g)$ are DDH tuples.

OT protocol with full security

- R (with input j) chooses random a_0,a_1,r in [1,q]. Computes
  - $h_0=g^{a0}$, $h_1=g^{a1}$, $a=g^r$, and $b_0=(h_0)^r g^j$, $b_1=(h_1)^r g^j$ and sends them.
- Let $h=h_0/h_1$, and $b=b_0/b_1$. The receiver proves that (g,h,a,b) is a Diffie-Hellman tuple with relation to (G,q).

- If sender accepts the proof, it chooses random u_0,v_0,u_1,v_1 in [1,q] and computes and sends
  - $w_0=a^{u0}g^{v0}$, and encryption of x_0 with the key $z_0=(b_0)^{u0}(h_0)^{v0}$.
  - $w_1=a^{u1}g^{v1}$, and encryption of x_1 with the key $z_1=(b_1/g)^{u1}(h_1)^{v1}$.

- Receiver decrypts z_j with the key $(w_j)^{a_j}$

Correctness when both parties are honest

- If j=0
  - $(w_0)^{a_0} =$
  - $(a^{u0} g^{a0}\ g^{v0 a0}) =$
  - $(g^{u0\ a0\ r}\ g^{v0\ a0}) = g^{u0\ r\ a0}\ g^{v0\ a0} = b_0{}^{u0}(h_0)^{v0} = z_0$

- If j=1
  - $(w_1)^{a_1} =$
  - $(a^{u1} g^{a1}\ g^{v1 a1}) =$
  - $(g^{u1\ a1\ r}\ g^{v1\ a1}) = g^{u1\ r\ a1}\ g^{v1\ a1}\ x_1 = (b_1/g)^{u1}(h_1)^{v1} = z_1.$

Security proof for a corrupt sender

- We construct a simulator interacting with the sender
  - Like an honest R, Sim computes $h_0=g^{a0}$, $h_1=g^{a1}$.
  - Unlike an honest R, Sim computes $a=g^r$, $b_0=(h_0)^r$, $b_1=(h_1)^r g$. (The issue here is that g is multiplied into b_1 but not into b_0.)
  - Sim sends these values to the sender.
  - Sim "cheats" in the zero-knowledge proof to convince the sender that the values it sent are legitimate.
    - Namely, runs the proof. Saves the state of the sender. Learns what challenge the sender asks, and then goes back to the saved state and continues from that point with messages that agree with the challenge.

Security proof for a corrupt sender

- We construct a simulator interacting with the sender
  - Like an honest R, Sim computes $h_0=g^{a0}$, $h_1=g^{a1}$.
  - Unlike an honest R, Sim computes $a=g^r$, $b_0=(h_0)^r$, $b_1=(h_1)^r g$. (The issue here is that g is multiplied into b_1 but not into b_0.)
  - Sim sends these values to the sender.
  - Sim "cheats" in the zero-knowledge proof to convince the sender that the values it sent are legitimate.
  - Sim receives two encryptions from the sender. Since it cheated, it can decrypt **both** of them as $x_0=z_0/(w_0)^{a_0}$ and $x_1=z_1/(w_1)^{a_1}$.
  - Sim sends x_0,x_1 to the trusted party computing the OT functionality.

Security proof for a corrupt sender (cont)

- The only difference in sender's view between the simulation and a real run is that in the simulation it receives $b_0=(h_0)^r, b_1=(h_1)^r g$ whereas in the real run it receives $b_0=(h_0)^r g^b, b_1=(h_1)^r g^b$.
- We must show that if the DDH holds then the following two tuples are indistinguishable
  - $h_0=g^{a0}, h_1=g^{a1}, a=g^r, b_0=(h_0)^r, b_1=(h_1)^r g$
  - $h_0=g^{a0}, h_1=g^{a1}, a=g^r, b_0=(h_0)^r, b_1=(h_1)^r$ (this is for b=0)
- We will show that these tuples are indistinguishable if it is hard to distinguish between tuples of the form (g,h,g^r,h^r) and $(g,h,g^r,h^r g)$. (This is equivalent to DDH assumption.)

Security proof for a corrupt sender (cont)

- We must show that if it is hard to distinguish between (g,h,g^r,h^r) and (g,h,g^r,h^rg) then the following two tuples are indistinguishable
  1. $h_0=g^{a0}, h_1=g^{a1}, a=g^r, b_0=(h_0)^r, b_1=(h_1)^rg$
  2. $h_0=g^{a0}, h_1=g^{a1}, a=g^r, b_0=(h_0)^r, b_1=(h_1)^r$ (this is for b=0)
- Suppose that distinguishing between Cases 1 and 2 is easy
  - We receive a challenge $(g,h,s=g^r,t)$ and need to find out if $t=h^r$ or $t=h^rg$.
  - Choose random a_0 and compute $h_0=g^{a0}, b_0=s^{a0}$, and $h_1=h, b_1=t$. If $t=h^r$ then this is exactly as in Case 2, and if $t=h^rg$ then this is exactly as in Case 1.
  - Therefore distinguishing the cases solves this DDH variant.

Security proof for a corrupt receiver

- We construct a simulator interacting with the receiver
  - Sim receives (h_0,h_1,a,b_0,b_1) from the corrupt receiver. If they are not all in the group then Sim aborts.
  - Sim "cheats" in the zero-knowledge proof.
    - (In this proof the receiver proves that $(g, h=h_0/h_1, g^r, b=b_0/b_1)$ is a Diffie-Hellman tuple, by proving that it knows the corresponding r.)
    - Namely, Sims saves the state of the receiver, rewinds the protocol and runs it again, and using the information it learned in both runs it extracts r.
  - Sim computes $c=b_0/(h_0)^r$. If $c=1$ then Sim sets $j=0$. Otherwise it sets $j=1$.

Security proof for a corrupt receiver (cont)

- We construct a simulator interacting with the receiver
  - Sim sends j to the trusted party computing the OT functionality and receives back x_b.
  - Sim computes (w_0,z_0) and (w_1,z_1) like an honest sender.
  - Sim encrypts x_j using the key z_j, and encrypts x_{1-j} using the key z_{1-j}.
  - Sim outputs whatever the corrupt receiver outputs, and halts.

Security proof for a corrupt receiver (cont)

- Analysis
  - We need to show that the distributions of A's view in the real run and in the simulations are indistinguishable.
  - The only difference between the two views is the way in which the input x_{1-j} is encrypted.
  - We prove separately for the case where $c=b_0/(h_0)^r$ equals 1, equals g, or equals a different value.
- c=1 (in this case Sim sets j=0)
  - Due to the ZK proof, there is an r s.t. $a=g^r$ and $b_0/b_1=(h_0/h_1)^r$. Since $c=b_0/(h_0)^r=1$, then $b_1=h_1^r$.
  - x_1 is encrypted using $w_1=(g^r)^{u1}g^{v1}$ and $z_1=(b_1/g)^{u1}(h_1)^{v1}$. Since b_1,h_1,u_1,v_1 are distributed as in the real run, the distributions are equal.

Security proof for a corrupt receiver (cont)

- Analysis (cont.)
- c=g (in this case Sim sets j=1)
  - Again, due to the ZK proof, there is an r s.t. $a=g^r$ and $b_0=gh_1^r$.
  - x_0 is encrypted using $w_0=(g^r)^{u0}g^{v0}$ and $z_0=(b_0)^{u0}(h_0)^{v0}$. Since b_0,h_0,u_0,v_0 are distributed as in the real run, the distributions are equal.
- c has a different value (in this case Sim sets j=1)
  - There is an r s.t. $a=g^r$ and $b_0=g^th_1^r$,for some value d.
  - x_0 is encrypted using $w_0=(g^r)^{u0}g^{v0}$ and $z_0=(b_0)^{u0}(h_0)^{v0}=(h_1)^{u0}(g^t)^{u0}(h_0)^{v0}$
  - This is equivalent to setting $u'_0=u_0+1/t$ and $v'_0=v_0-r/t$.
  - Now $b_0,h_0,u'_0,v;_0$ are distributed as in the real run.