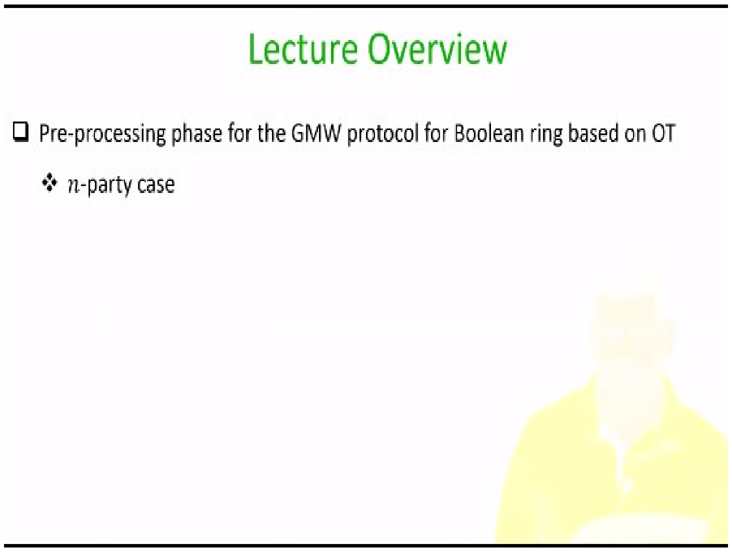
Secure Computation: Part 1

Prof. Ashish Choudhury
Department of Computer Science & Engineering
International Institute of Information Technology-Bengaluru

Lecture - 43
Pre-Processing Phase for the GMW Protocol: The n-Party Case

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:32)

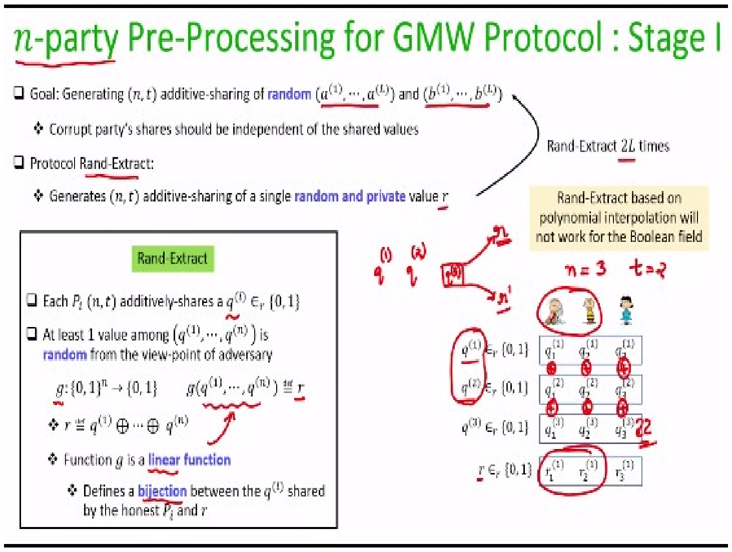


Lecture Overview

- Pre-processing phase for the GMW protocol for Boolean ring based on OT
  - n-party case

So in the last lecture we have seen the pre-processing phase for the GMW protocol for the Boolean ring case. And we saw the 2-party case based on OTs. We will now extend that method for the n-party case and the extension is very trivial, okay.

(Refer Slide Time: 00:48)



n-party Pre-Processing for GMW Protocol : Stage I

- Goal: Generating (n, t) additive-sharing of random $(a^{(1)}, \cdots, a^{(L)})$ and $(b^{(1)}, \cdots, b^{(L)})$
  - Corrupt party's shares should be independent of the shared values
- Protocol Rand-Extract:
  - Generates (n, t) additive-sharing of a single random and private value r

Rand-Extract $2L$ times

Rand-Extract based on polynomial interpolation will not work for the Boolean field

Rand-Extract

- Each P_i (n, t) additively-shares a $q^{(i)} \in_r \{0, 1\}$
- At least 1 value among $(q^{(1)}, \cdots, q^{(n)})$ is random from the view-point of adversary

$g: \{0, 1\}^n \rightarrow \{0, 1\}$ $\quad g(q^{(1)}, \cdots, q^{(n)}) \stackrel{\text{def}}{=} r$

- $r \stackrel{\text{def}}{=} q^{(1)} \oplus \cdots \oplus q^{(n)}$
- Function g is a linear function
  - Defines a bijection between the $q^{(i)}$ shared by the honest P_i and r

$q^{(1)}$ $q^{(2)}$ $\boxed{q^{(3)}}$
n
n'
$n = 3$ $t = 2$
$q^{(1)} \in_r \{0,1\}$ $q_1^{(1)}$ $q_2^{(1)}$ $q_3^{(1)}$
$q^{(2)} \in_r \{0,1\}$ $q_1^{(2)}$ $q_2^{(2)}$ $q_3^{(2)}$
$q^{(3)} \in_r \{0,1\}$ $q_1^{(3)}$ $q_2^{(3)}$ $q_3^{(3)}$
$r \in_r \{0,1\}$ $r_1^{(1)}$ $r_2^{(1)}$ $r_3^{(1)}$

So for the n-party case, our goal is again divided into two stages. In stage I we have to generate additive secret sharing for the bits, a 1 to a L, b 1 to b L where the bits a 1 to a L and b 1 to b L should be uniformly random and every party should have a bit share for this bits so that the corrupt party's shares does not reveal anything about the underlying shared bits, okay.

And again, this can be done by just extending the randomness extraction procedure that we had discussed in the last lecture to the n-party case. So this Rand-Extract procedure will generate a secret sharing for one random bit, we can run it 2L times to generate 2L number of random bits in a secret shared fashion. And again I demonstrate, let me demonstrate this process assuming n is equal to 3 and t is equal to 2.

Each party will now act as a dealer and secret share a random bit. So P 1 acts as a dealer, secret shares q 1. P 2 acts as a dealer, secret shares the bit q 2. P 3 acts as a dealer and secret shares a bit q 3 which is random. Now what we can say is that since there is at least one honest party among these n parties, so for simplicity assume that P 1 and P 2 are the bad guys. So they know the full values q 1 and q 2.

But they do not know what is the bit q 3 shared by the third party because that is coming from the privacy property of your additive secret sharing. For the third bit P 1 and P 2 together jointly holds two shares, but the third share is missing for them. It could be any random share and hence it could be any random bit q 3 shared by the third party.

So to generate or randomly extract a uniformly random bit, we define a linear function g which takes as inputs the bits shared by the respective parties namely the q bits which are respectively shared by the parties P 1, P 2, P n. And output is defined to be the XOR of the input bits, this is a linear function.

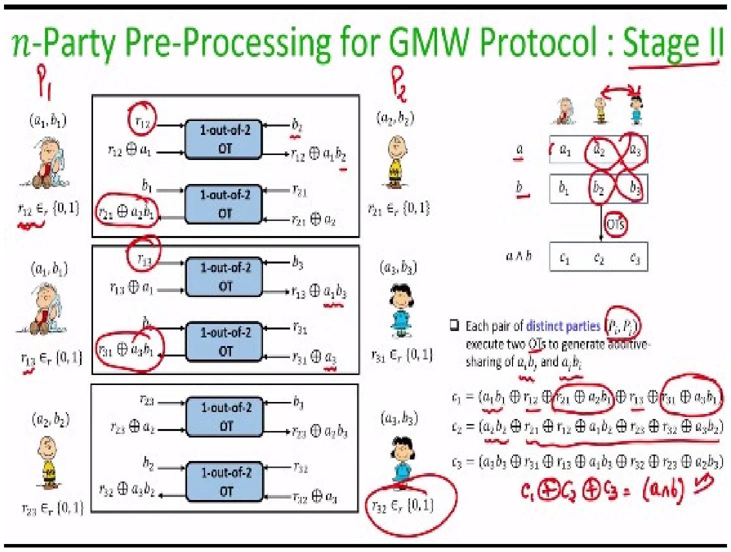
And since it is a linear function, it can be applied on the shares of the input values and that will produce the shares of the output values. And this can be done locally. So that means if each party just exhort their respective shares of all the q values that will give them respectively their shares of the r value.

And my claim is that this value r this bit r is now a uniformly random bit from the viewpoint of the two corrupt parties or n – 1 corrupt parties. This is because this here is a very nice bijection which is defined by the function g between the q bit shared by the honest party and the resultant output r. What does that mean? Again, taking this example since the first two parties are corrupt here, they know the values q 1 and q 2, okay.

So they know q 1, the bit q 1, they know the full bit q 2, but they do not know what is the full bit q 3. It could be either 0 or 1. So if it is 0 then that leads to a candidate r. If it is 1 it leads to a candidate r prime. But since q 3 is picked uniformly at random, the resultant r could be 0 or 1 with equal probability.

And the two corrupt parties would not be knowing what exactly is the output r which is now finally secret shared because for the value r they will be only having two shares, okay. So that will be the stage I for the n-party case.

(Refer Slide Time: 04:46)



n-Party Pre-Processing for GMW Protocol : Stage II

- Each pair of distinct parties (P_i, P_j) execute two OTs to generate additive-sharing of a_ib_j and a_jb_i

$$c_1 = (a_1b_1 \oplus r_{12} \oplus r_{21} \oplus a_2b_1 \oplus r_{13} \oplus r_{31} \oplus a_3b_1)$$
$$c_2 = (a_2b_2 \oplus r_{21} \oplus r_{12} \oplus a_1b_2 \oplus r_{23} \oplus r_{32} \oplus a_3b_2)$$
$$c_3 = (a_3b_3 \oplus r_{31} \oplus r_{13} \oplus a_1b_3 \oplus r_{32} \oplus r_{23} \oplus a_2b_3)$$
$$c_1 \oplus c_2 \oplus c_3 = (a \wedge b)$$

Now we have to focus on stage II. And now for the stage II again, the goal will be to do the following. You have the a component and b component of all the L multiplication triples which are secret shared, that has been done already in stage I. We have to now run OT instances among the n parties so that the and of the a and b component is also made available in a secret shared fashion among the parties.

But in the process, nothing additional about the bit a and the bit b should be revealed. And the idea here will be a very simple generalization of the idea which we had for the 2-party case. In the 2-party case there were two cross terms which were the source of problem and we somehow securely computed a secret sharing of those two cross terms itself based on OT instances. But now we are in the n-party case.

And here between every pair of parties P i and P j where i is different from j, there are two cross terms, namely a i times b j, and a j times b i, right? And we have to somehow securely compute these two cross terms by executing the OT instances. So in general, if we have n parties, there will be order of n square number of cross terms.

And for every pair of cross terms involving every pair of two distinct parties, we will run two OT instances to get securely to securely compute those two cross terms. So let me demonstrate what I am saying here. So between 1 and 2, P 1 and P 2, we have the cross terms, a 1 times b 2, and a 2 times b 1. We cannot afford to reveal a 1 or b 1 to the second party. In the same way, we cannot afford to reveal a 2 or b 2 to the first party.

But somehow we want to securely compute a secret sharing of a 1 times b 2 and a 2 times b 1 only among these two parties. How we can do that? Just run the OT based idea that we had discussed in the last lecture for the 2-party case, assuming that there is no third party and our cross terms are only a 1 times b 2 and a 2 times b 1. Namely the first party, it acts as a center in one of the OT instances by masking his a share.

And the second party is acting as a receiver with b 2 as his choice bit. Whereas the second party acts as a sender in the other OT instance by masking is a share with a random mask and the first party he acts as a receiver with his b share as the choice bit. And as a result of the correctness of the OT instances, the second party will receive a masking of a 1 times b 2.

The first party will now receive a masking of a 2 times b 1. So that will take care of the cross terms involving P 1 and P 2. In the same way let us focus on the cross terms involving the second party and the third party assuming there is no first party. And there are now two cross terms involving these two parties. So run the similar idea here. Oh sorry, I am demonstrating right now the first and the third party, sorry.

So between the first and the third party okay, so a 1 and b 3 are involved, that is one cross term. And a 3 and b 1 is involved. So again, P 1 acts as a sender masking his a share. And now that this masking is done using an independent pad which is different from the pad which was used to mask his share of a when he interacted with the second party, right?

And in the same way the third party acts as a sender in one of the OT instances with her share of a and masking it with a random pad, okay. And now based on the choice spades, the cross term, a 1 times b 3 goes in a masked fashion to the third party. That cross term a 3 times b 1 goes in a masked fashion to the first party.

And now there will be another OT instance involving the second and the third party to take care of the cross terms a 2 times b 3 and a 3 times b 2 okay, where again independent masks are used. And now the overall share of a times b for the three parties will be as follows. So there will be something which they can locally compute namely a i b i. So P i can compute a i b i.

And now P i will do the following. For all the OT instances where he acted as a sender, right so he acted as a sender in two OT instances. So this OT instance he set this as a mask and this OT instance he set as a mask. So he will XOR them. And he acted as a receiver in two OT instances. So whatever he received in those two OT instances, the XOR term. That will be his overall share for the and of a and b.

And same logic P 2, first compute a 2 b 2 and then whatever the mask he has picked while acting as the sender for OT instances and whatever are the masked cross terms he has received as a receiver in the OT, instances everything masked together that will be c 2 and similarly for P 3. And now you can see here that indeed c 1 XORed with c 2 XORed with c 3 gives you the and of a and b.

And the security of the bits a and b are preserved because for each independent OT instances random masks are used for masking the a shares or the b shares, sorry for masking the a shares and the privacy of the shares of b is maintained because that comes from the privacy of the receiver’s choice bit.

And hence you can now see that the pre-processing phase for the, stage II of the pre-processing phase for the GMW protocol based on the OT instances how naturally it extends the n-party case. You do not have to do anything sophisticated here. Thank you.