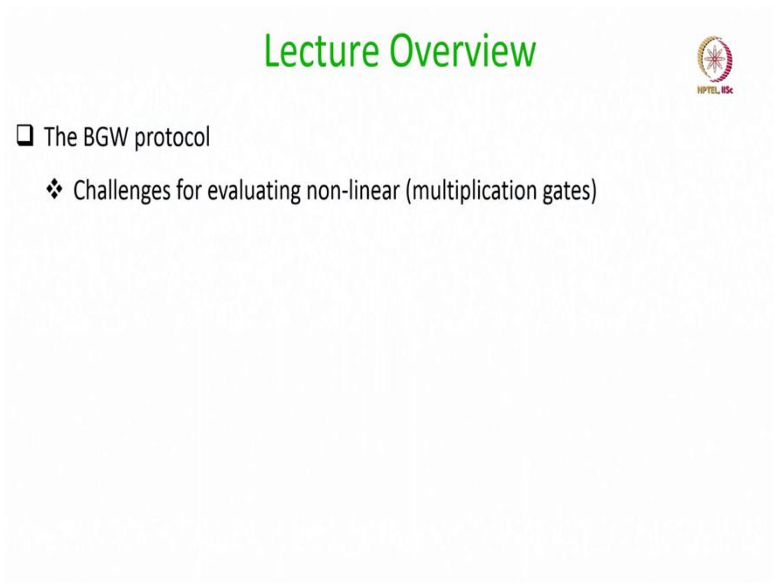
Secure Computation: Part II

Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 46
The BGW MPC Protocol: The Case of Non-Linear Gates

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:25)

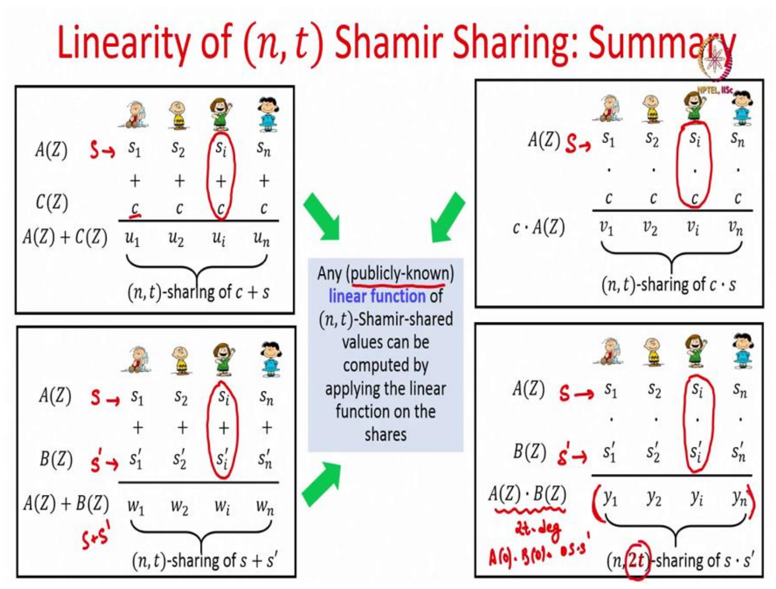


Lecture Overview

- The BGW protocol
  - Challenges for evaluating non-linear (multiplication gates)

So, in this lecture we will discuss what are the challenges we face when we try to run the BGW protocol for evaluating the non-linear gates in the circuit namely the multiplication gates.

(Refer Slide Time: 00:37)



Linearity of (n, t) Shamir Sharing: Summary

$A(Z)$ $s \rightarrow$ s_1 s_2 s_i s_n; $+$; $C(Z)$ c c c c; $A(Z) + C(Z)$ u_1 u_2 u_i u_n — (n, t)-sharing of $c + s$

$A(Z)$ $s \rightarrow$ s_1 s_2 s_i s_n; $\cdot$; c c c c; $c \cdot A(Z)$ v_1 v_2 v_i v_n — (n, t)-sharing of $c \cdot s$

$A(Z)$ $s \rightarrow$ s_1 s_2 s_i s_n; $+$; $B(Z)$ $s' \rightarrow$ s'_1 s'_2 s'_i s'_n; $A(Z) + B(Z)$ w_1 w_2 w_i w_n — $s+s'$ — (n, t)-sharing of $s + s'$

$A(Z)$ $s \rightarrow$ s_1 s_2 s_i s_n; $\cdot$; $B(Z)$ $s' \rightarrow$ s'_1 s'_2 s'_i s'_n; $A(Z) \cdot B(Z)$ $(y_1$ y_2 y_i $y_n)$ — 2t-deg — $A(0) \cdot B(0) = s \cdot s'$ — $(n, 2t)$-sharing of $s \cdot s'$

Any (publicly-known) linear function of (n, t)-Shamir-shared values can be computed by applying the linear function on the shares

So, before that let us quickly go through the summary of the linearity property of Shamir Secret Sharing. So, we had seen earlier that Shamir Secret Sharing has this very nice linearity property that if you have secret shared inputs then you can compute any publicly known linear function of those secret shared inputs without requiring any interaction among the parties.

So, for instance, if you have 2 secrets say s and s’ which are secret shared through an instance of (n, t) Shamir secret sharing and if you want the parties to have their shares for s + s’ then it is enough if each party individually adds its shares of s and s’, that will give that party its share of s + s’ ok.

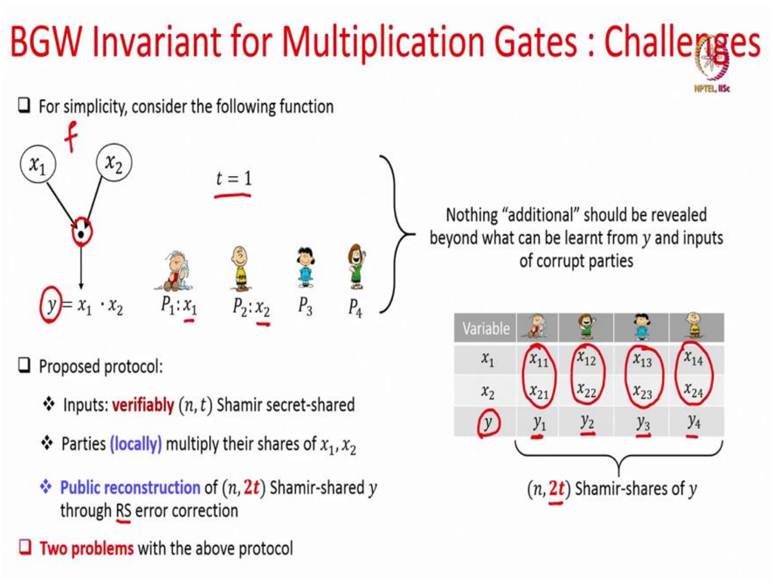
In the same way if there is some publicly known constant say c and if there is a value, if there is a secret s which is secret shared then to get a secret sharing of s plus c, it is enough if each party adds this constant c to its own share of s. And in the same way to get a secret sharing of c times s, it is enough if each party locally multiplies its share of s with this publicly known constant c.

However, if there are 2 secrets s and s prime which are secret shared as per the Shamir secret sharing scheme with degree of sharing being t, and now if each party locally multiplies its individual shares of s and s prime then the resultant vector of shares will lie on a 2 t degree polynomial whose constant term will be s times s prime. So, this polynomial

A times B is a 2 t degree polynomial and its constant term will be the product of the constant term of the A polynomial and B polynomial which is s times s prime.

So, even though this vector of shares constitutes a Shamir secret sharing of s times s prime, the problem here is that now the degree of the sharing is 2t, it is no longer t.

(Refer Slide Time: 03:20)



BGW Invariant for Multiplication Gates : Challenges

- For simplicity, consider the following function

x_1
x_2
f
$y = x_1 \cdot x_2$
$t = 1$
$P_1 : x_1$
$P_2 : x_2$
P_3
P_4
NPTEL, IISc

Nothing "additional" should be revealed beyond what can be learnt from y and inputs of corrupt parties

Variable				
x_1	x_{11}	x_{12}	x_{13}	x_{14}
x_2	x_{21}	x_{22}	x_{23}	x_{24}
y	y_1	y_2	y_3	y_4

$(n, 2t)$ Shamir-shares of y

- Proposed protocol:
  - Inputs: **verifiably** (n, t) Shamir secret-shared
  - Parties **(locally)** multiply their shares of x_1, x_2
  - **Public reconstruction** of $(n, 2t)$ Shamir-shared y through RS error correction
- **Two problems** with the above protocol

And that is 1 of the major problems which we face when we try to maintain the BGW invariant in the BGW protocol while evaluating the multiplication gates. So, now, we will discuss the challenges which we face when we try to maintain the BGW invariant for multiplication gates.

So, for simplicity consider that we have a simple function consisting of just a single multiplication gate in the circuit. So, you have 2 inputs x1 and x2 held by P1 and P2. P3 and P4 have no inputs here. So, you can imagine that their inputs are some public constant say 1 ok and the circuit output is the product of the inputs of party 1 and party 2. And for the purpose of demonstration here I am assuming t is equal to 1; that means, any 1 out of these 4 parties can be corrupt.

Now, we want the BGW protocol should help us to achieve the following property. We want at the end of the protocol everyone should learn the product of x1 and x2 and no additional information should be learnt beyond what can be learnt from the output y and the inputs of the corrupt parties.

So, what will be the BGW protocol in this context? Say every party verifiably secret shares is input using the polynomial based verifiable secret sharing schemes. In this case P1 will act as a dealer and invoke an instance of VSS to secret share its input x1, similarly P2 will act as a dealer to verifiably secret share its input x2. P3 and P4 does not have any inputs here ok.

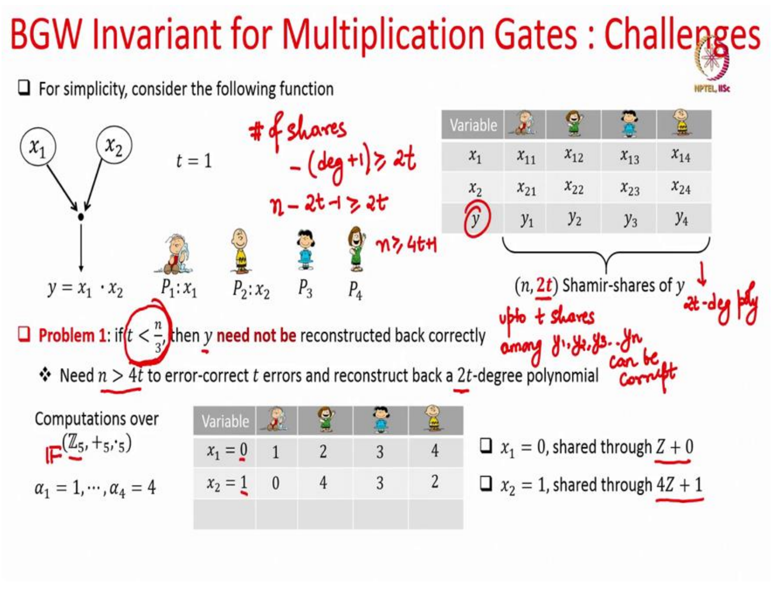
So, the shares of x1 are denoted by x11, x12, x13, x14 and one of the shares among these 4 shares will be available with one of the parties and so, on. Now suppose here that the parties try to go and evaluate this multiplication gate by simply multiplying their shares of x1 and x2 ok, that is what the parties would have done if this would have been an addition gate.

So, we are proposing the same steps even for a multiplication gate. So, the proposed protocol here is that if you have a multiplication gate in the circuit then let the parties multiply their respective shares of x1 and x2.

So, P1 multiplies its shares of x1 and x2 to get the share y1 and similarly P2 multiplies its share of x1 and x2 to get its share of y, P3 multiplies its shares of x1 and x2 to get its share of y and P4 multiplies its share of x1 and x2 to get its share of y. So, as I said earlier that this vector of shares y1, y2, y3, y4, they lie on a 2t degree polynomial now whose constant term is y.

And now since there are no more gates in the circuit. So, this is the function f which we want to compute right. So, as per the BGW protocol, the parties make public their shares of y and to error correct the potentially corrupt shares which can be made public by corrupt parties we apply the Reed Solomon error correction algorithm and hope that we get back the function output y, that is the proposed protocol. Now, there are two problems with the above protocol and let us discuss both problems individually.

(Refer Slide Time: 07:18)



BGW Invariant for Multiplication Gates : Challenges

- For simplicity, consider the following function

$$\begin{tikzcd} x_1 \arrow[dr] & & x_2 \arrow[dl] \\ & \bullet \arrow[d] & \\ & y = x_1 \cdot x_2 & \end{tikzcd}$$

$t = 1$

$P_1 : x_1$ $P_2 : x_2$ P_3 P_4

# of shares
− (deg + 1) ≥ 2t
n − 2t − 1 ≥ 2t
n ≥ 4t+1

Variable				
x_1	x_{11}	x_{12}	x_{13}	x_{14}
x_2	x_{21}	x_{22}	x_{23}	x_{24}
y	y_1	y_2	y_3	y_4

$(n, \underline{2t})$ Shamir-shares of y

2t-deg poly
upto t shares among y1, y2, y3 .. yn can be corrupt

- **Problem 1**: if $t < \frac{n}{3}$, then y **need not be** reconstructed back correctly
  - Need $\underline{n > 4t}$ to error-correct t errors and reconstruct back a $\underline{2t}$-degree polynomial

Computations over $\mathbb{F}(\mathbb{Z}_5, +_5, \cdot_5)$

$\alpha_1 = 1, \cdots, \alpha_4 = 4$

Variable				
$x_1 = 0$	1	2	3	4
$x_2 = 1$	0	4	3	2

- $x_1 = 0$, shared through $\underline{Z + 0}$
- $x_2 = 1$, shared through $\underline{4Z + 1}$

So, the first problem with the proposed protocol is that if we are in a setting where t < n/3 and which will be the case looking ahead for the BGW protocol, and which also happens to be the optimal resilience for perfectly secure MPC, then this function output y need not be reconstructed back correctly if t shareholders produce incorrect shares of y. This is because now the degree of the underlying sharing polynomial is 2t. Specifically we need the condition n > 4t to hold to error correct t errors and reconstruct back a 2t degree sharing polynomial right.

So, recall that the shares y1, y2, y3, y4 they now lie on a 2t degree polynomial and up to t shares among y 1, y 2, y 3 up to y n can be corrupt ok. So, for the purpose of demonstration we have taken t equal to one; that means, 1 out of these 4 shares can be corrupt which share is corrupt we do not know because we do not know the identity of the corrupt party. So, with the help of 4 shares among which 1 is corrupt and such that the shares are lying on a 2-degree polynomial, we will not be able to recover back the value y ok.

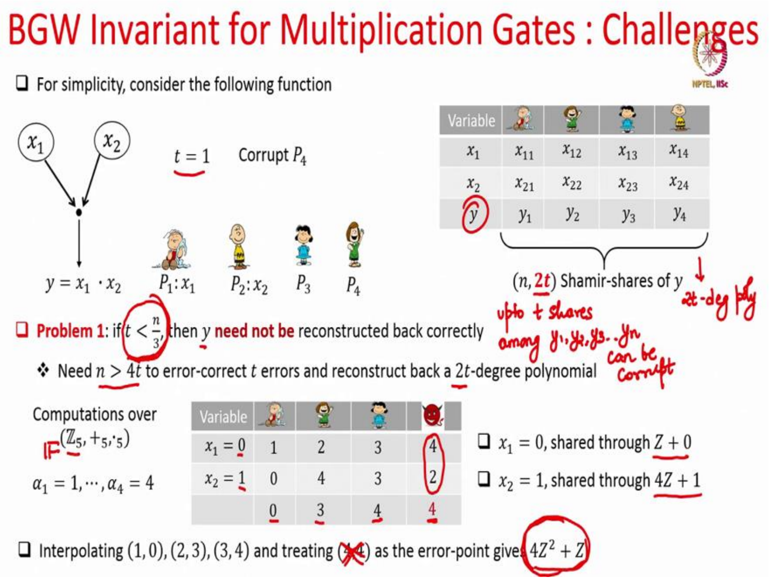
Because coding theory bound says that the number of shares minus degree of the polynomial plus 1 should be greater than equal to 2 times the number of errors which we want to correct error correct. So, in this case the number of shares is n ok and the degree of the polynomial happens to be now 2 t because these shares y1, y2, y3, y4 they constitute distinct points on a 2t degree polynomial ok. So, we need this condition to hold; that means, n should be greater than equal to 4t plus 1.

But that need not be the case because we would like to design the BGW protocol and looking ahead we will see that the actual BGW protocol is with only t less than n over 3. So, if we are in the setting where t is less than n over 3 and if we want to reconstruct a 2t degree polynomial using the help of n shares of which up to t are corrupt we will not be able to do that and as a result the parties may end up reconstructing an incorrect function output.

So, let me demonstrate this problem with an example where I take my field as Z5 and my evaluation points associated with the 4 parties are 1, 2, 3 and 4. So, imagine x1 is equal to 0 and x 2 is equal to 1 and say x 1 is VSS shared through this 1-degree polynomial resulting in the shares 1, 2, 3 and 4 for the parties P 1, P 2, P 3 and P 4 respectively, remember that all the plus and all the multiplication operations are performed modulo 5 here.

In the same way suppose the value x 2 is equal to 1 and is VSS shared through this 1-degree polynomial, this will result in the shares 0, 4, 3 and 2 for the first, second, third and fourth party respectively.

(Refer Slide Time: 11:24)



BGW Invariant for Multiplication Gates : Challenges

- For simplicity, consider the following function

$$\begin{tikzcd} x_1 \arrow[dr] & & x_2 \arrow[dl] \\ & \bullet \arrow[d] & \\ & y = x_1 \cdot x_2 & \end{tikzcd}$$

$\underline{t = 1}$ Corrupt P_4

$P_1 : x_1$ $P_2 : x_2$ P_3 P_4

Variable				
x_1	x_{11}	x_{12}	x_{13}	x_{14}
x_2	x_{21}	x_{22}	x_{23}	x_{24}
y	y_1	y_2	y_3	y_4

$(n, \underline{2t})$ Shamir-shares of y

2t-deg poly

upto t shares among $y_1, y_2, y_3 .. y_n$ can be corrupt

- **Problem 1**: if $t < \frac{n}{3}$, then y **need not be** reconstructed back correctly
  - Need $\underline{n > 4t}$ to error-correct t errors and reconstruct back a $2t$-degree polynomial

Computations over $\mathbb{F}$ $(\mathbb{Z}_5, +_5, \cdot_5)$

$\alpha_1 = 1, \cdots, \alpha_4 = 4$

Variable				
$x_1 = \underline{0}$	1	2	3	4
$x_2 = \underline{1}$	0	4	3	2
	$\underline{0}$	$\underline{3}$	$\underline{4}$	$\underline{4}$

- $x_1 = 0$, shared through $\underline{Z + 0}$
- $x_2 = 1$, shared through $\underline{4Z + 1}$

- Interpolating $(1, 0), (2, 3), (3, 4)$ and treating $(\cancel{4, 4})$ as the error-point gives $4Z^2 + Z$

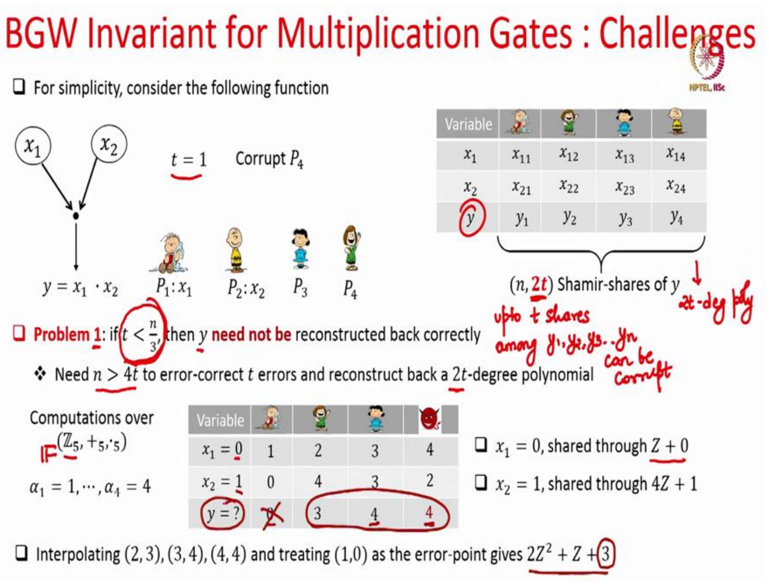
And now since one of the 4 parties can be corrupt imagine that it is the fourth party ok. Of course, this is just for the demonstration purpose, parties will not be knowing who exactly is the correct party except that one among the 4 parties is correct.

So, as per the proposed protocol every party should locally multiply its shares of x1 and x2 and make that share public. So, P1 will make public the share 0, P2 is honest it will make public the share 3, P3 is also honest it will make public the share 4, but P 4 ideally should make public the share 3 because the product of its shares of x 1 and x 2 is 8; 8 modulo 5 is 3, but since P 4 is corrupt it is making an incorrect share say the value 4 as public.

Now, everyone knows that among these 4 shares one of the shares is corrupt and these shares lie on a 2-degree polynomial. Now, if the parties ignore the share of the fourth party ok and interpolate the shares of the first second and third party, then they will reconstruct a 2-degree polynomial $4Z^2 + Z$ whose constant term is 0 resulting in the function output y is equal to 0 and actually this is the right output, this is the supposed polynomial, which the parties should reconstruct if they have the knowledge that party 4 is the corrupt party.

So, if the parties would have this prior knowledge that it is the fourth party who is corrupt, then they could have simply ignored the share of the last party, the fourth party and by interpolating the remaining 3 shares they would have recovered the right polynomial and the right output. But in this case the parties have absolutely no idea who exactly is the corrupt entity among the 4 entities.

(Refer Slide Time: 13:58)



BGW Invariant for Multiplication Gates : Challenges

- For simplicity, consider the following function

$$\begin{tikzcd} x_1 \arrow[dr] & & x_2 \arrow[dl] \\ & \bullet \arrow[d] & \\ & y = x_1 \cdot x_2 & \end{tikzcd}$$

$t = 1$ Corrupt P_4

$P_1 : x_1$ $P_2 : x_2$ P_3 P_4

Variable				
x_1	x_{11}	x_{12}	x_{13}	x_{14}
x_2	x_{21}	x_{22}	x_{23}	x_{24}
y	y_1	y_2	y_3	y_4

$(n, 2t)$ Shamir-shares of y

2t-deg poly

upto t shares among $y_1, y_2, y_3 \ldots y_n$ can be corrupt

- **Problem 1**: if $t < \frac{n}{3}$, then y **need not be** reconstructed back correctly
  - Need $n > 4t$ to error-correct t errors and reconstruct back a $2t$-degree polynomial

Computations over $\mathbb{F}(\mathbb{Z}_5, +_5, \cdot_5)$

$\alpha_1 = 1, \cdots, \alpha_4 = 4$

Variable				
$x_1 = 0$	1	2	3	4
$x_2 = 1$	0	4	3	2
$y = ?$	$\cancel{0}$	3	4	4

- $x_1 = 0$, shared through $Z + 0$
- $x_2 = 1$, shared through $4Z + 1$

- Interpolating (2, 3), (3, 4), (4, 4) and treating (1,0) as the error-point gives $2Z^2 + Z + 3$

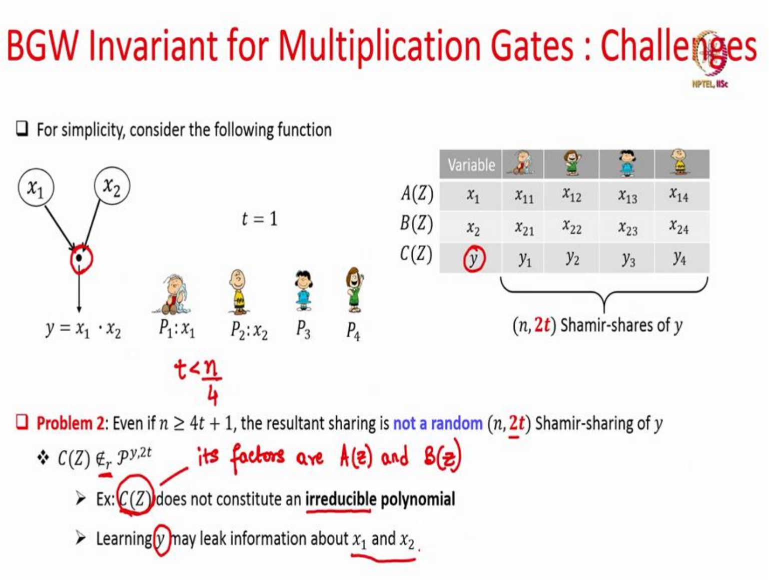
So, this is just one heuristic, it could be also possible that parties make the heuristic that it is the first party who is corrupt. So, ignore the share of the first party and use the remaining

3 shares and see what exactly is the 2-degree polynomial which they get back and in this case they will be getting back this polynomial and they might end up taking 3 as the function output.

So, as you can see here that the error correction completely fails here. Parties have absolutely no idea which share to ignore and what are the remaining 3 shares they should consider for reconstructing the function output, depending upon which 3 shares they take, they will be getting different possible outputs and as a result they will be completely clueless which output to consider as the right output.

So, that is precisely the problem number 1 associated while maintaining the BGW invariant. Namely, if we let the parties simply multiply their respective shares of the multiplication gate inputs then that will cause a problem because the number of shares that we require now to reconstruct the function output after error correcting t errors is less, ok, because the degree of the sharing for the function gate output is now no longer t it is now a 2 t degree polynomial ok. So, that is the first problem associated here.

(Refer Slide Time: 15:33)



BGW Invariant for Multiplication Gates : Challenges

NPTEL, IISc

- For simplicity, consider the following function

x_1
x_2
$y = x_1 \cdot x_2$
$t = 1$
$P_1 : x_1$
$P_2 : x_2$
P_3
P_4
$t < \frac{n}{4}$

	Variable				
$A(Z)$	x_1	x_{11}	x_{12}	x_{13}	x_{14}
$B(Z)$	x_2	x_{21}	x_{22}	x_{23}	x_{24}
$C(Z)$	y	y_1	y_2	y_3	y_4

$(n, 2t)$ Shamir-shares of y

- **Problem 2**: Even if $n \geq 4t + 1$, the resultant sharing is not a random $(n, 2t)$ Shamir-sharing of y
  - $C(Z) \notin_r \mathcal{P}^{y,2t}$ its factors are A(z) and B(z)
    - Ex: $C(Z)$ does not constitute an **irreducible polynomial**
    - Learning y may leak information about x_1 and x_2.

Now, the second problem is a more serious problem, this is related to the privacy here. So, for the moment imagine that we are working in a setting where t is less than n over 4 we are taking still the same function namely a function where we will have just 1 multiplication gate and now our proposed protocol should work, there should not be any error in the correctness property because we are now assuming that we have t less than n over 4.

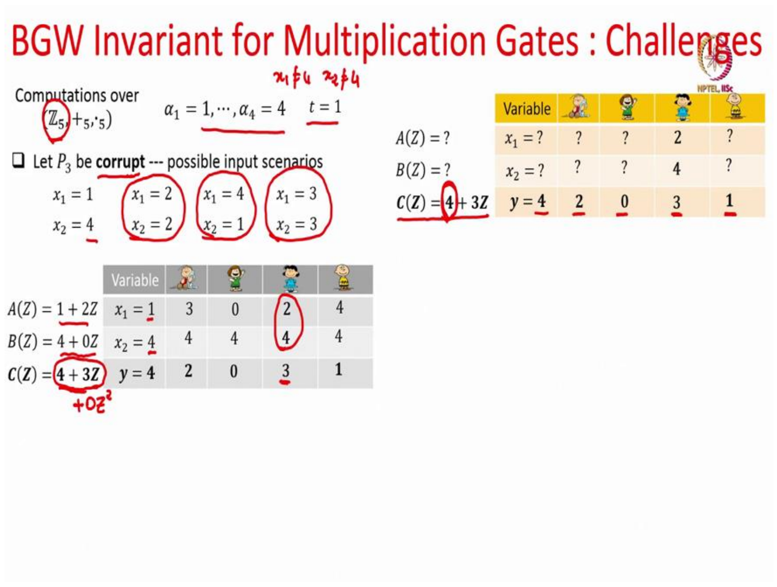
So, even if up to t incorrect shares are provided for the value y, we have sufficient redundancy, the Reed Solomon error correction algorithm has sufficient redundancy to error correct those t errors and correctly recover back the 2 t degree polynomial through which the value y is secret shared.

However, the problem now is that this polynomial C(Z) which is a 2t degree polynomial is no longer a random polynomial ok. So, it is not the case that all the coefficients of the C polynomial, except of course, the constant term are random coefficients ok. Whereas, the BGW invariant requires that if the gate inputs are secret shared, for maintaining the privacy we require the output of the gate to be secret shared through a random t degree polynomial, but that is not happening here.

Because this C polynomial is no longer a random polynomial its coefficients are not random. For instance this C polynomial does not constitute an irreducible polynomial because its factors are the A polynomial and the B polynomial whereas, if C is a random 2t degree polynomial then with equal probability it could have been a reducible polynomial and with equal probability it could have been an irreducible polynomial, but as per the proposed protocol we are landing up in a case where the C polynomial is always going to be an irreducible polynomial and hence it's not a random 2 t degree polynomial.

Consequently, in the proposed protocol if we let the parties simply learn the function output y by making by asking the parties to make their shares of y namely the shares lying on these C polynomial public, then it may end up leaking information about the inputs of the parties and which simply goes against the privacy property.

(Refer Slide Time: 18:44)



BGW Invariant for Multiplication Gates : Challenges

Computations over $(\mathbb{Z}_5, +_5, \cdot_5)$ $\quad \alpha_1 = 1, \cdots, \alpha_4 = 4 \quad t = 1$

$x_1 \neq 4 \quad x_2 \neq 4$

- Let P_3 be **corrupt** --- possible input scenarios

$x_1 = 1$	$x_1 = 2$	$x_1 = 4$	$x_1 = 3$
$x_2 = 4$	$x_2 = 2$	$x_2 = 1$	$x_2 = 3$

	Variable				
$A(Z) = 1 + 2Z$	$x_1 = 1$	3	0	2	4
$B(Z) = 4 + 0Z$	$x_2 = 4$	4	4	4	4
$C(Z) = 4 + 3Z$ $+0Z^2$	$y = 4$	2	0	3	1

	Variable				
$A(Z) = ?$	$x_1 = ?$	?	?	2	?
$B(Z) = ?$	$x_2 = ?$	?	?	4	?
$C(Z) = 4 + 3Z$	$y = 4$	2	0	3	1

NPTEL, IISc

So, again let me demonstrate this problem with an example here. So, my field for demonstration purpose remains the same my evaluation points associated with the parties are 1, 2, 3 and 4 and here one out of the 4 parties could be corrupt imagine that the input of the first party is 1 which is VSS shared through the Shamir sharing polynomial 1 plus 2 Z resulting in the shares 3, 0, 2 and 4 for the first, second, third and fourth party respectively

Imagine that the second input is 4 which is VSS shared through the Shamir sharing polynomial 4 plus 0 z resulting in the share 4 for all the 4 parties and suppose the parties locally multiply their shares of x 1 and x 2. And imagine for simplicity here that there is no malicious corruption here, everyone behaves honestly in the sense that even the corrupt party makes public the right share of y ok.

So, every party is making here public the right share of y, you can see here and now by interpolating the shares here, interpolating the shares of y which have been made public, the parties will reconstruct a 2-degree polynomial 4 + 3Z. You might be wondering it is not a 2-degree polynomial, well its a 2 degree polynomial where the coefficient of z square is 0 and then they will take 4 as the function output.

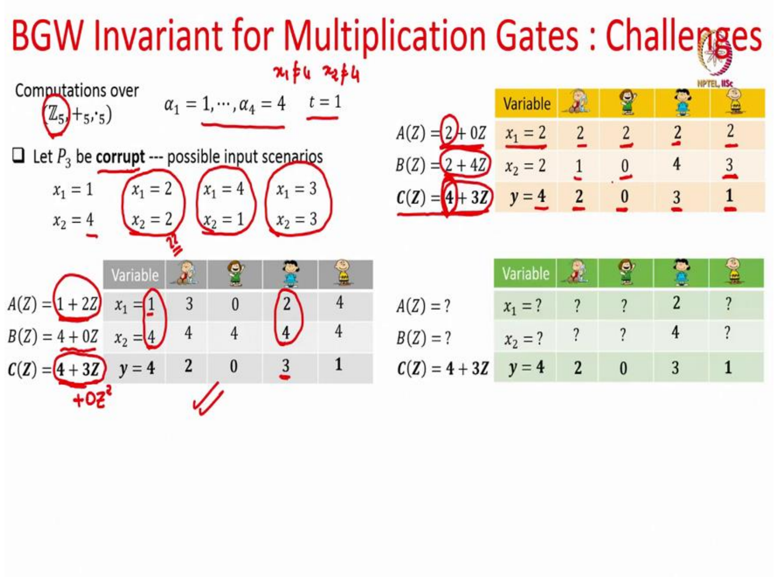
So, now imagine P 3 is corrupt the third part is corrupt, but it is corrupt in a passive way it is not maliciously corrupt. So, you can see here that the party P 3 is under the control of the adversary, but it has made public the right share. So, that is why in this example there are no incorrect shares for the value y at the time of reconstructing y due to which we are able to reconstruct the function output. So, you might be wondering that this protocol is maintaining the privacy property, but I am going to show that that is not the case.

Now, since P 3 is corrupt what exactly it learns? It learns the function output and it learns all the shares of y which are made public by the individual parties and hence the corresponding 2-degree sharing polynomial through which the value y has been secret shared, namely the C polynomial and all the values which are question mark here are the unknown values from the viewpoint of the party P 3.

Now, party 3 also knows that since the function output is 4, the possible input scenarios are the following: either the first input is 1 and the second input is 4 or the first input was 2 and the second input was also 2 or the first input was 4 and the second input was 1 or the first input was 3 and second input was 3. There is no other possibility because the inputs can take any value from the set 0 to 4 ok. So, for instance x 1 cannot be 4 and x 2 cannot be 4 because x 1 being 4 and x 2 being 4 should result in an output 1, but the output that parties learn is here in this case is 4.

Now, what we are going to show here is that, there are certain possibilities for x 1 and x 2 which the corrupt P 3 can rule out based on the values which it has learnt in this execution and this simply implies a violation of the perfect privacy because if the proposed protocol for evaluating the multiplication gate is perfectly secured then the view of the corrupt P 3 namely whatever values the corrupt P 3 learns in the protocol which are marked or which are shown here in bold it should not lead the corrupt P 3 to conclude what could be the possible x 1, x 2 and what could not be the possible x 1, x 2 ok.

(Refer Slide Time: 23:21)



BGW Invariant for Multiplication Gates : Challenges

Computations over $(\mathbb{Z}_5, +_5, \cdot_5)$ $\alpha_1 = 1, \cdots, \alpha_4 = 4$ $t = 1$

$x_1 \neq 4$ $x_2 \neq 4$

- Let P_3 be **corrupt** --- possible input scenarios

$x_1 = 1$	$x_1 = 2$	$x_1 = 4$	$x_1 = 3$
$x_2 = 4$	$x_2 = 2$	$x_2 = 1$	$x_2 = 3$

$2Z$

	Variable				
$A(Z) = 1 + 2Z$	$x_1 = 1$	3	0	2	4
$B(Z) = 4 + 0Z$	$x_2 = 4$	4	4	4	4
$C(Z) = 4 + 3Z$ $+0Z^2$	$y = 4$	2	0	3	1

	Variable				
$A(Z) = 2 + 0Z$	$x_1 = 2$	2	2	2	2
$B(Z) = 2 + 4Z$	$x_2 = 2$	1	0	4	3
$C(Z) = 4 + 3Z$	$y = 4$	2	0	3	1

	Variable				
$A(Z) = ?$	$x_1 = ?$	?	?	2	?
$B(Z) = ?$	$x_2 = ?$	?	?	4	?
$C(Z) = 4 + 3Z$	$y = 4$	2	0	3	1

So, for instance suppose the corrupt P3 makes the hypothesis that is it possible that P1's input was 2, it makes this hypothesis. Now in order that P1's input is 2 and the share which the third party has received corresponding to that input is 2, it must be the case that party 1 has secret shared its input using the Shamir sharing polynomial 2 + 0z. That was not actually the case, actually P 1 has secret shared the input 1 using the sharing polynomial 1+ 2 Z.

But right now what we are doing is, we are just doing some hypothesis on the behalf of the corrupt P3. So, corrupt P3 is thinking in its mind, is it possible that P1 has participated in the protocol with x1 equal to 2? And the answer is yes provided P1 sends 2 to the first, second and fourth party.

And now since the input x1 has been fixed to 2; that means, adversary is basically checking this particular combination. So, since x1 has been fixed to 2, x2 should also be 2. Now it is quite possible that x2 was 2 namely the input of the second party was 2 and the second party has given 4 as the share to the third party provided his sharing polynomial was 2 + 4 z in which case P2 should have distributed the shares 1, 0 and 3 to the first second and fourth party respectively.

And now you can see magically what is happening here is that if P1's share would have been 2 and 1 for x 1 and x 2, then it would have made public the share 2. If P2's share would have been 2 and 0 for x 1 and x 2 then it would have made the share 0 for y public.

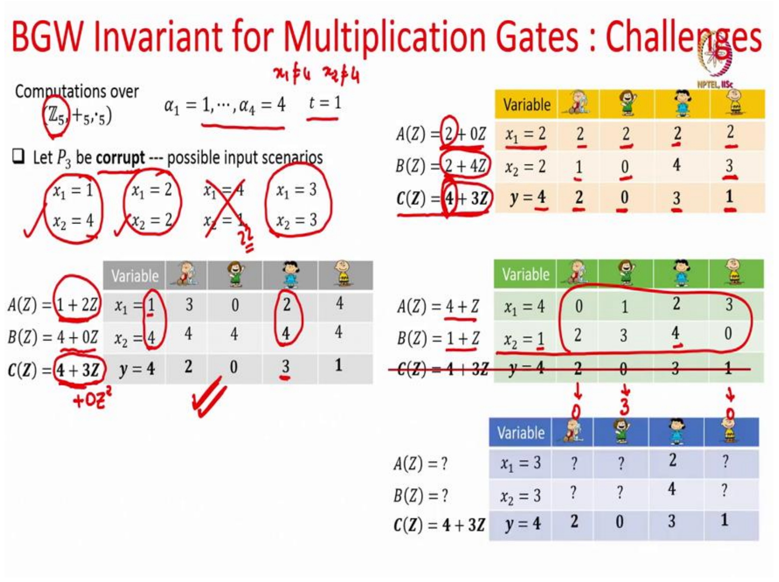
And if P4’s shares are 2 and 3 then it would have made public the share 1 for the value y. And all these shares would have led to the interpolation of the c polynomial 4 + 3 Z.

So, that means, adversary cannot rule out the possibility of x 1 being 2 and x 2 being 2. Namely based on the values which a corrupt P 3 has seen in the protocol execution, so it has seen the values in the protocol execution corresponding to 1 and 4 because that were the real inputs of P1 and P2, but what I have shown in this table is that the view of the corrupt P3 is also consistent with x1 being 2 and x2 being 2.

Namely, it is quite possible that there is some randomness for P1 and P2 such that whatever the values P 3 has seen in this execution, the same values P 3 would have seen if P 1 would have participated with input 2 and P 2 would have participated with input 2 right.

So; that means, 2 of the possibilities adversary cannot rule out. So, x1 being 4 and x2 being 4 is possible and x1 being 2 and x2 being 2 is also possible.

(Refer Slide Time: 27:12)



BGW Invariant for Multiplication Gates : Challenges

Computations over $(\mathbb{Z}_5, +_5, \cdot_5)$ $\alpha_1 = 1, \cdots, \alpha_4 = 4$ $t = 1$

$x_1 \neq 4$ $x_2 \neq 4$

❑ Let P_3 be **corrupt** --- possible input scenarios

$x_1 = 1$, $x_2 = 4$ ✓ $x_1 = 2$, $x_2 = 2$ ✓ ~~$x_1 = 4$, $x_2 = 1$~~ 22 $x_1 = 3$, $x_2 = 3$

	Variable				
$A(Z) = 1 + 2Z$	$x_1 = 1$	3	0	2	4
$B(Z) = 4 + 0Z$	$x_2 = 4$	4	4	4	4
$C(Z) = 4 + 3Z$ $+0Z^2$	$y = 4$	2	0	3	1

✓✓

	Variable				
$A(Z) = 2 + 0Z$	$x_1 = 2$	2	2	2	2
$B(Z) = 2 + 4Z$	$x_2 = 2$	1	0	4	3
$C(Z) = 4 + 3Z$	$y = 4$	2	0	3	1

	Variable				
$A(Z) = 4 + Z$	$x_1 = 4$	0	1	2	3
$B(Z) = 1 + Z$	$x_2 = 1$	2	3	4	0
~~$C(Z) = 4 + 3Z$~~	~~$y = 4$~~	~~2~~ $\downarrow$ 0	~~0~~ $\downarrow$ 3	~~3~~	~~1~~ $\downarrow$ 0

	Variable				
$A(Z) = ?$	$x_1 = 3$	?	?	2	?
$B(Z) = ?$	$x_2 = 3$	?	?	4	?
$C(Z) = 4 + 3Z$	$y = 4$	2	0	3	1

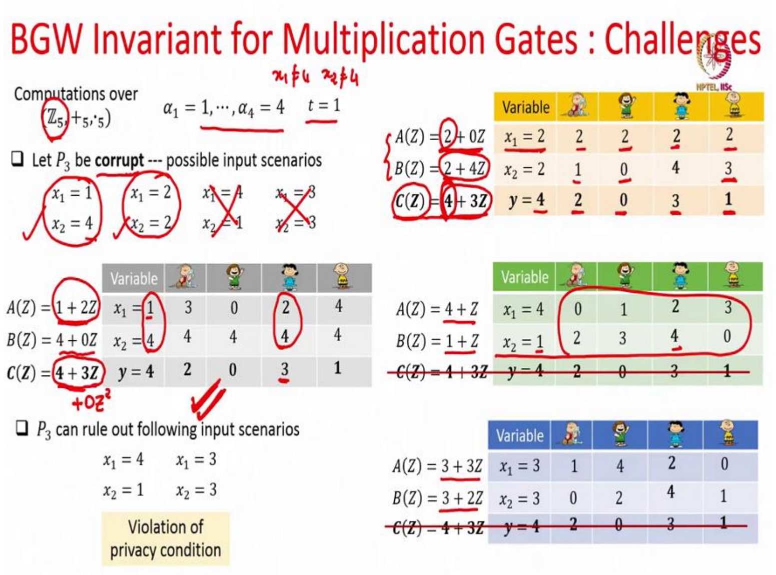
Now adversary is trying to see whether it could be possible that x1 was 4 and x2 was 1, given that the bold values are the values which adversary has seen in the protocol execution.

Now, if x1 was 4 and if P3’s share was 2 and if P1’s sharing polynomial was a 1 degree polynomial, then that could be possible only if the sharing polynomial of P1 was 4 + Z leading to the shares 0, 1 and 3 for the first party, second party and fourth party respectively. And the second party’s input being 1, with P3’s share being 4 is possible only if P2 has used the 1 degree polynomial 1 + Z for secret sharing its input.

But now you can see that if these are the shares of x1 and x2, then that will not lead to the shares of y being 2, 0, 3 and 1. Because if P1’s shares for x1 was 0 and for x2 was 2, then P1 should have made the share 0 public for the value y, P2 should have made 3 public as its share of y and P4 should have made public 0 as the share for y, but they have made public the shares 2, 3 and 1 respectively.

So that means, now adversary can rule out this possibility; that means, based on the values which are tick marked here which it has actually seen in the protocol it can say that no, its not the case that I have participated in the protocol where P1’s input was 4 and P2’s input was 1. And in the same way now let us check the fourth possibility whether x1 being 3 and x2 being 3 is possible.

(Refer Slide Time: 29:17)



BGW Invariant for Multiplication Gates : Challenges

Computations over $(\mathbb{Z}_5, +_5, \cdot_5)$ $\alpha_1 = 1, \cdots, \alpha_4 = 4$ $t = 1$

$x_1 \neq 4$ $x_2 \neq 4$

- Let P_3 be **corrupt** --- possible input scenarios

$x_1 = 1$, $x_2 = 4$ ✓ $x_1 = 2$, $x_2 = 2$ ✓ ~~$x_1 = 4$, $x_2 = 1$~~ ~~$x_1 = 3$, $x_2 = 3$~~

	Variable				
$A(Z) = 1 + 2Z$	$x_1 = 1$	3	0	2	4
$B(Z) = 4 + 0Z$	$x_2 = 4$	4	4	4	4
$C(Z) = 4 + 3Z$	$y = 4$	2	0	3	1

$+0Z^2$

✓

- P_3 can rule out following input scenarios

$x_1 = 4$ $x_1 = 3$

$x_2 = 1$ $x_2 = 3$

Violation of privacy condition

	Variable				
$A(Z) = 2 + 0Z$	$x_1 = 2$	2	2	2	2
$B(Z) = 2 + 4Z$	$x_2 = 2$	1	0	4	3
$C(Z) = 4 + 3Z$	$y = 4$	2	0	3	1

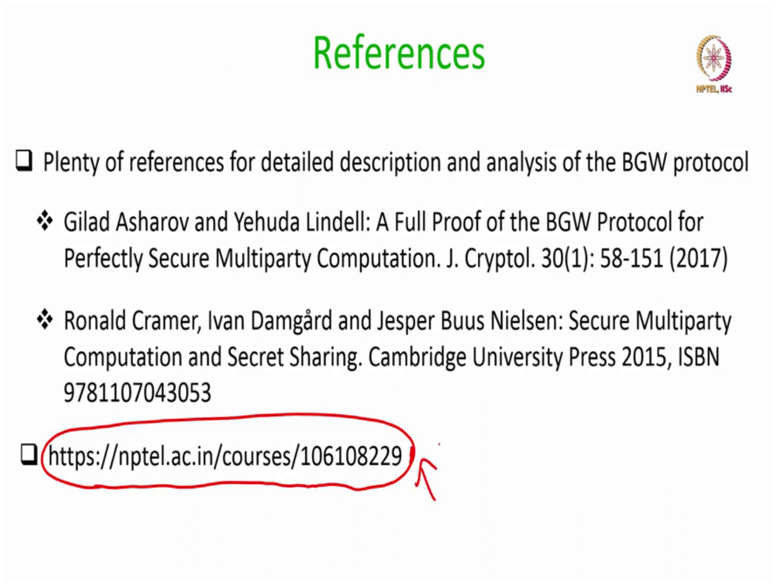
	Variable				
$A(Z) = 4 + Z$	$x_1 = 4$	0	1	2	3
$B(Z) = 1 + Z$	$x_2 = 1$	2	3	4	0
~~$C(Z) = 4 + 3Z$~~	~~$y = 4$~~	~~2~~	~~0~~	~~3~~	~~1~~

	Variable				
$A(Z) = 3 + 3Z$	$x_1 = 3$	1	4	2	0
$B(Z) = 3 + 2Z$	$x_2 = 3$	0	2	4	1
~~$C(Z) = 4 + 3Z$~~	~~$y = 4$~~	~~2~~	~~0~~	~~3~~	~~1~~

Well, in that case the P1’s sharing polynomial should have been 3 + 3z. And P2’s sharing polynomial should have been 3 + 2z. And again in that case it turns out that adversary can rule out the possibility of the first and the second input being 3 because that is not going to be consistent with the shares of y, which the corrupt party sees in the protocol while reconstructing the function output.

So, that means, there are 2 possible input scenarios which the adversary can rule out here. And that is a violation of the privacy condition. And the second problem is arising here because the C polynomial is not a random 2-degree polynomial, it is for instance reducible polynomial whose factors are the A polynomial and the B polynomial.

(Refer Slide Time: 30:17)



References

- Plenty of references for detailed description and analysis of the BGW protocol
  - Gilad Asharov and Yehuda Lindell: A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation. J. Cryptol. 30(1): 58-151 (2017)
  - Ronald Cramer, Ivan Damgård and Jesper Buus Nielsen: Secure Multiparty Computation and Secret Sharing. Cambridge University Press 2015, ISBN 9781107043053
- https://nptel.ac.in/courses/106108229

So, with that I end this lecture. To summarize we have seen here the challenges associated while evaluating a multiplication gate for maintaining the BGW gate invariant.

In the next lecture we will see what exactly we have to do to deal with the challenges. And these are the references which are used for today’s lecture. So, in the first course secure computation part 1, we have a lot of discussion regarding the challenges associated while evaluating the multiplication gate. So, most of my explanation in today’s lecture is taken from that course.

Thank you.