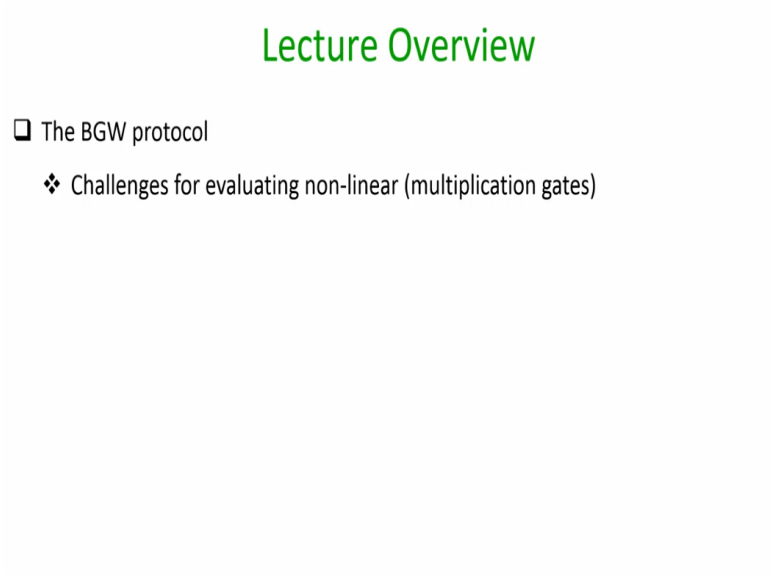
Secure Computation - Part I
Prof. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology, Bangalore

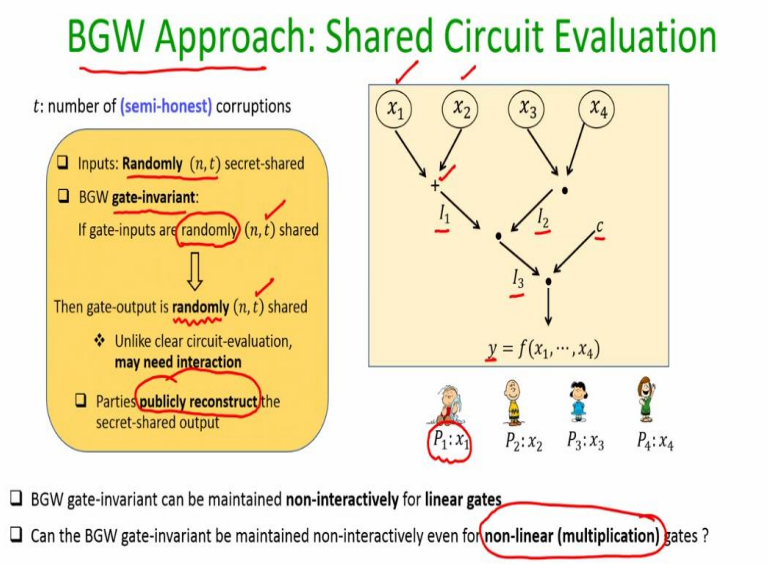
Module - 4
Lecture - 22
The BGW MPC Protocol: The Case of Non-Linear Gates

(Refer Slide Time: 00:32)



Hello everyone. Welcome to this lecture. So, the plan for this lecture is as follows: In this lecture, we will see the description of BGW MPC protocol where we will now consider we also have non-linear gates which are also known as multiplication gates in the circuit. And then, we will see what the challenges which we face when we try to securely evaluate the multiplication gates in the circuit are.

(Refer Slide Time: 00:55)



So, to quickly recap, this is the BGW approach for shared circuit-evaluation where we assume that the function which needs to be securely computed is represented by a publicly-known arithmetic circuit consisting of linear gates and non-linear gates. The shared circuit-evaluation starts with secret-sharing the respective inputs of the parties. This is done by the parties themselves.

Each party acts as a dealer and secret-shares its input with the degree of sharing being t. And the important thing is that the instance of secret-sharing is random; in the sense that, if P_1 is the dealer and if it wants to secret-share the value x_1, it will be generating random shares for this value x_1 and distributing the shares and so on. And then, once all the input values are secret-shared, each party starts evaluating a copy of the circuit, not on the clear value, but rather on the shares of the value.

So, in more detail, parties try to maintain what we call as the BGW gate-invariant, where it will be ensured that, if there is a gate whose inputs are secret-shared randomly with the degree of sharing being t, then the gate-output is made available to the parties in a secret-shared fashion, where the degree of sharing is still t and the vector of shares is a random vector of shares.

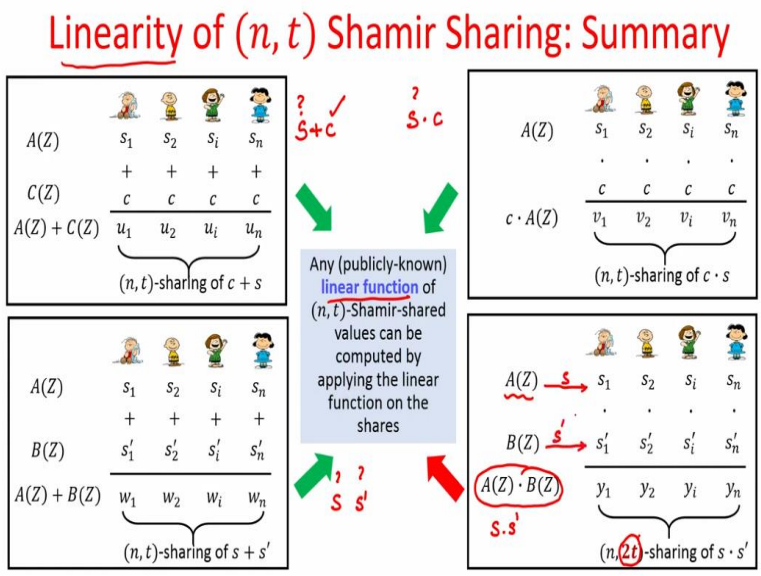
And in this whole process, neither the gate-inputs nor the gate-output will be revealed. So, for instance, consider this input gate. I am saying that once the values x_1 and x_2 are secret-shared, the parties will try to ensure that they have their shares of this intermediate I_1. And then, the parties will ensure that they have the shares of this intermediate value I_2.

And then, they will ensure that, given the shares of I_1 and I_2, the parties own their respective shares of I_3. And then, this c is a publicly known constant, so, the parties will ensure that, given the shares of I_3, they obtain their respective shares of y. And finally, once all the values, intermediate values in the circuit have been made available in a secret-shared fashion, the parties go and publicly reconstruct the final output value.

I stated earlier that to maintain this gate-invariant, the parties may need to interact. We had seen that the (n, t) secret-sharing scheme which is used in the BGW protocol is the Shamir secret-sharing scheme. And since Shamir secret-sharing satisfies the linearity property to maintain the gate-invariant for linear gates, it does not require any interaction among the parties. But we will see that, when it comes to the multiplication gates, to maintain the gate-invariant, the parties need to interact.

So, the focus of this lecture will be to answer this question - can we maintain this BGW gate-invariant without asking the parties to interact even for the multiplication gates? And we will see that the answer is no.

(Refer Slide Time: 04:42)

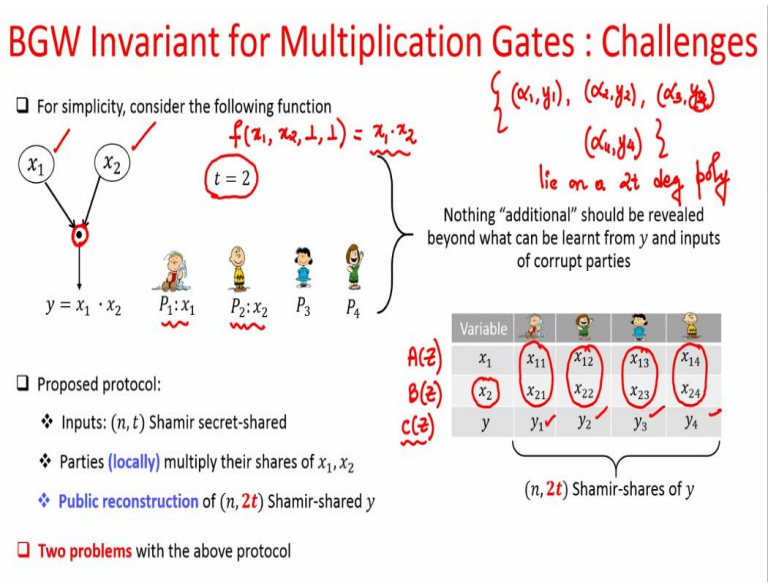


So, before going into that, here is a quick recap of the linearity property of (n, t) secret-sharing where we had seen that any linear function of secret-shared inputs can be computed by applying the function on the shares of the input values itself. So, for instance, if you want to compute $s + c$, where c is publicly-known but s is not known but rather secret-shared; then, by adding the shares of s with the value c, the parties obtain a secret-sharing of $s + c$.

In the same way, if s and s' are 2 unknown values, but secret-shared with the same threshold t, then, to obtain shares of $s + s'$, each party just has to go and add its respective share of s and s'. Similarly, if s is unknown and if c is a publicly-known value, and if each party multiplies its respective share of s with this public known constant c, then it obtain its respective share of $c \cdot s$.

But when it comes to multiplying 2 secret-shared values s and s', we had seen that if we ask the parties to just locally multiply their respective shares of s and s', then the resultant vector of shares will constitute a secret-sharing of$s \cdot s'$, but the degree of sharing will be $2t$ and not t; because the resultant vector of shares will now lie on a polynomial of the degree-$2t$ and not of degree t. Namely, this polynomial $A(Z) \cdot B(Z)$ will be a 2-degree polynomial, where $A(Z)$ is the polynomial for secret-sharing the value s and $B(Z)$ is the polynomial secret-sharing the value s'.

(Refer Slide Time: 06:49)



So, now, we will see what the challenges we face when we try to maintain the BGW invariant for evaluating the multiplication gates. And to demonstrate the challenges, I will consider a very simple scenario. I will consider a 4-party function where party 1 has the input x_1, party 2 has the input x_2, party 3 has no input and party 4 has no input, and all the 4 parties are interested to learn $x_1 \cdot x_2$, where x_1 and x_2 are some field elements.

I will assume that, out of these 4 parties, up to t parties can be passively corrupted by a computationally-unbounded adversary. The security goal is to ensure that the corrupt parties

should not learn anything additional beyond what they can learn from $x_1 \cdot x_2$ and the inputs of 2 corrupt parties which are under the control of the adversary. So, this is a very simple function consisting of just 1 multiplication gates.

There is no linear gate; there is just 1 multiplication gate, and you obtain the function output. So, that is our goal. Now, let us see what goes wrong if we try to blindly follow the BGW MPC protocol for this circuit, where we start with asking the input holders to secret-share their respective inputs; and then we multiply the shares of x_1 and x_2; and then we finally go and publicly reconstruct the vector of shares of $x_1 \cdot x_2$.

We will see what exactly goes wrong in this protocol. So, we start with the inputs and we asked P_1 to act as a dealer and P_2 to act as a dealer and independently secret-share their respective inputs x_1 and x_2 with random 2-degree polynomials. Here is the table of values. So, x_1 will be secret-shared and it will produce 4 shares. The first share will be with the first party, second share with the second party and so on.

Similarly, P_2 will act as a dealer and it will pick a 2-degree polynomial, a random 2-degree polynomial whose constant term is x_2, generate 4 shares and distribute the individual shares to the respective parties. This will complete the input sharing phase. There are no more inputs for this function. And now, parties start evaluating the circuit. So, they now have the inputs for this multiplication gate available in a secret-shared fashion.

They take the first gate in this circuit, which is the multiplication gate. Let the parties locally multiply their respective shares of x_1 and x_2. So, party 1 multiplies x_{11} with x_{21}. When I say multiply, multiply as per the field multiplication operation. And it will produce a field element; call it y_1. Similarly, P_2 goes and multiplies its respective shares of x_1 and x_2; P_3 multiplies its respective shares of x_1 and x_2; and P_4 multiplies its respective shares of x_1, x_2.

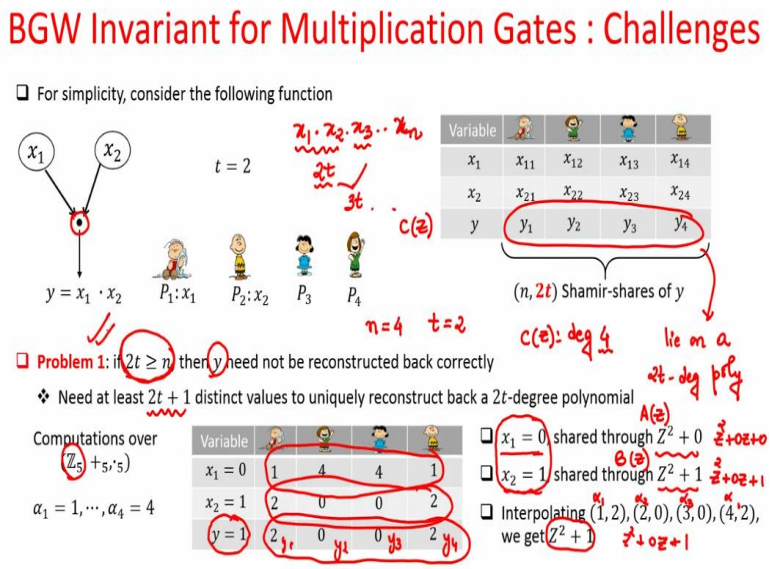
And let y_1, y_2, y_3, y_4 be the resultant vector of shares, where the i^{th} component is with the i^{th} party. So, as we have seen earlier, this vector of values y_1, y_2, y_3, y_4, namely, (α_1, y_1); (α_2, y_2); (α_3, y_3); and (α_4, y_4); they constitute or they lie on a $2t$-degree polynomial, because x_1 was shared through a polynomial A whose degree was t. B was secret-shared through a t-degree

polynomial; call it as a B polynomial. The constant term of A was x_1. The constant term of B polynomial was x_2.

And now, if you multiply these 2 polynomials, you will get a C polynomial, and its degree will be $2t$. And the C polynomial evaluated at $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, will be giving you y_1, y_2, y_3, y_4. So, that is why, this vector of shares y_1, y_2, y_3, y_4, it constitutes an $(n, 2t)$ Shamir sharing of y. And now, let as per the BGW protocol, since there are no more gates after this multiplication gates, the parties have evaluated all the gates and the function output is available in a secret-shared fashion; so, let the parties exchange the shares of y among themselves.

That means, y_1 is given to everyone, y_2 is given to everyone, y_3 is given to everyone and y_4 is given to everyone. And then they interpolate a 2-degree polynomial namely the C polynomial and get back the value 1, suppose this is the BGW protocol, which we use here to evaluate this circuit. Now, there are 2 problems with our protocol.

(Refer Slide Time: 12:47)



And one problem is associated with the correctness property and the other problem is associated with the privacy property. So, let us see the problem associated with correctness. So, consider a case where the number of parties n, satisfies this condition. That means, $n \leq 2t$. Now, if this is the case, then the problem that we are going to face in the proposed protocol is that, the value of y need not be reconstructed back correctly, because, this vector of sharesy_1, y_2, y_3, y_4, they lie on a $2t$-degree polynomial.

And to reconstruct a $2t$-degree polynomial through Lagrange's interpolation, we need at least $2t + 1$ distinct values or $2t + 1$ distinct shares on that polynomial, to uniquely reconstruct it back. But if we are in a setting where $n \leq 2t$, then, even if *all* the shares of y are made public, we *cannot* reconstruct back the polynomial $C(Z)$ correctly. And if polynomial $C(Z)$ is not reconstructed correctly, you cannot reconstruct back y correctly.

And hence, the parties may not obtain the correct output. So, to make my point more clear, here, we are actually, in this example, in the setting where the number of parties is 4 and your $t = 2$; and imagine that we take this specific example where $x_1 = 0$, and P_1 shares $x_1 = 0$ through a 2-degree polynomial. So, this is a 2-degree polynomial. You can interpret it as $Z^2 + 0 \cdot Z + 0$.

And this polynomial will be evaluated at $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, which I am taking to be 1, 2, 3 and 4 respectively. So, this will be your vector of shares for x_1. And say $x_2 = 1$. It is secret-shared through this polynomial. So, again, this polynomial, you can interpret it as $Z^2 + 0 \cdot Z + 11$. So, this is a 2-degree polynomial. This polynomial evaluated at 1, 2, 3, 4 will produce these vector of shares.

Now, let the parties multiply their respective shares of x_1 and x_2. Remember, all the computations are performed over $\mathbb{Z}_5$, where the additions are addition modulo 5 and multiplications are multiplication modulo 5. So, this will be your y_1, y_2, y_3, y_4. And now, if we interpolate the points y_1, y_2, y_3, y_4, of course, along with $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, we will be getting back this polynomial.

I am not showing the computations in the slide, but you can verify it. If you interpolate the points (α_1, y_1); (α_2, y_2); (α_3, y_3); and (α_4, y_4); you will be getting this polynomial. This polynomial is also a 2-degree polynomial, because I can interpret it as this. And now, this is the polynomial which parties reconstruct back. Then, the output of the function which parties will obtain is 1.

But that is not the correct output, because, if $x_1 = 0$ and if $x_2 = 1$, then y should be 0. But what parties are reconstructing here? They are reconstructing 1 as the function output. And why this problem is happening? This problem is happening because the A polynomial was this

$Z^2 + 0$; the B polynomial was $Z^2 + 1$; and now, if I consider these 4 values y_1, y_2, y_3, y_4, they lie on the C polynomial.

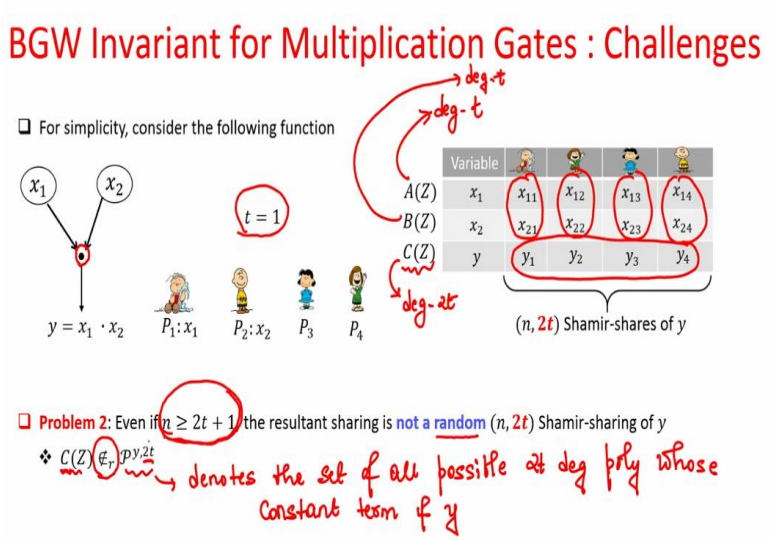
And C polynomial has degree-4, because it is the product of A polynomial and B polynomial, each of which is of degree-2; and if you multiply 2 polynomials of degree-2, you will get a resultant polynomial whose degree is 4. And to reconstruct uniquely and correctly a 4-degree polynomial, you need 5 distinct points or 5 y values or y shares on that C polynomial, but we are having only 4 parties in the system.

You do not have the fifth party who can provide you the fifth point on this C polynomial in the form of its share, and using which you can reconstruct back your y correctly. So, that is a correctness problem. And you can imagine that this problem is arriving just for 1 single multiplication gate; but imagine a circuit where you have a computation of the form $x_1 \cdot x_2 \cdot \ldots \cdot x_n$, a computation of this form.

So, you started, suppose you follow the same protocol there, where all the inputs are secret-shared; and then, say you ask the parties to locally multiply their shares of x_1 and x_2; the degree of sharing becomes $2t$. And now, you ask the parties to take this vector of shares and locally multiply their respective shares from this vector with the shares of x_3, then the resultant degree will become $3t$.

And like that, the degree of the resultant output will keep on increasing after every multiplication. And finally, the degree may become so high that you do not have sufficient number of parties to provide those many shares during the reconstruction of that final output. So, that will lead to an error in the correctness property. So, that is the problem number 1 or challenge number 1.

(Refer Slide Time: 19:14)



Now, let us see the second problem, second challenge associated with evaluating the multiplication gates; and this is related to the privacy property; and this is very subtle to understand. So, imagine for the time being that you are in a setting where $n \geq 2t + 1$. That means, I take, say for instance, the same example, same setting where n = 4, but now, I am working in a setting where t = 1.

It is given to me that up to 1 party can be semi-honestly corrupted, my passively corrupted by an unbounded adversary. That means, we now, no longer face that issue of correctness or problem 1 which we had encountered just now. Why so? Because, now we have just 1 multiplication gate. If x_1 is secret-shared through a degree-t polynomial A and if B is also secret-shared through a random degree-t polynomial, then my C polynomial will be of degree-$2t$.

And I am assuming that I am in a setting where$n \geq 2t + 1$. That means, if parties make public their respective shares of y, we have now sufficient number of shares to uniquely get back this C polynomial and take its constant term as the final output. So, correctness is no longer an issue, because I am ensuring that we are in the setting where n is greater than equal to $2t + 1$.

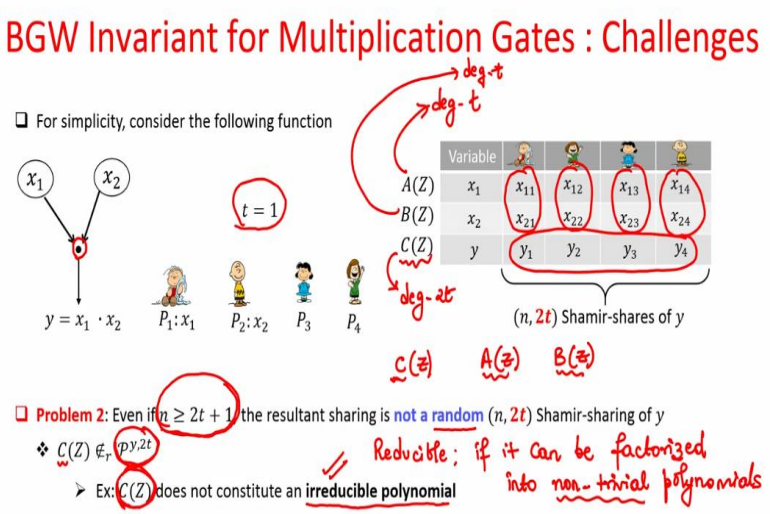
But now, the problem is that the resultant shares of y which are obtained by locally multiplying the shares of x; even though the degree of sharing is $2t$, the resultant vector of y shares do not constitute a random vector of shares for the value y lying on a $2t$-degree polynomial. What does that mean? I mean here that, if you take this $C(Z)$ polynomial and (α_1, y_1); (α_2, y_2);

(α_3, y_3); and (α_4, y_4) lying on the $C(Z)$ polynomial; of course the degree of C polynomial is $2t$; I can no longer claim that C polynomial could be a random polynomial from the set of all possible polynomials of degree-$2t$ whose constant term is y.

So, remember, this set $\mathcal{P}^{(y,2t)}$ denotes the set of all possible $2t$-degree polynomial whose constant term is y. So, my claim is, the C polynomial over which these values (α_1, y_1); (α_2, y_2); (α_3, y_3); and (α_4, y_4) lie, is not a randomly chosen element from this set, or it is not a randomly chosen polynomial whose constant term is y and whose degree is $2t$. What does it mean?

When I say that I am picking a random polynomial whose constant term is y and whose degree is $2t$, by that I mean that each of the coefficients of that polynomial except the constant term is randomly chosen; but the C polynomial that we are obtaining here is not a polynomial whose each of the coefficients except the constant coefficient or the constant term is randomly chosen from the field. Why so?

(Refer Slide Time: 23:27)



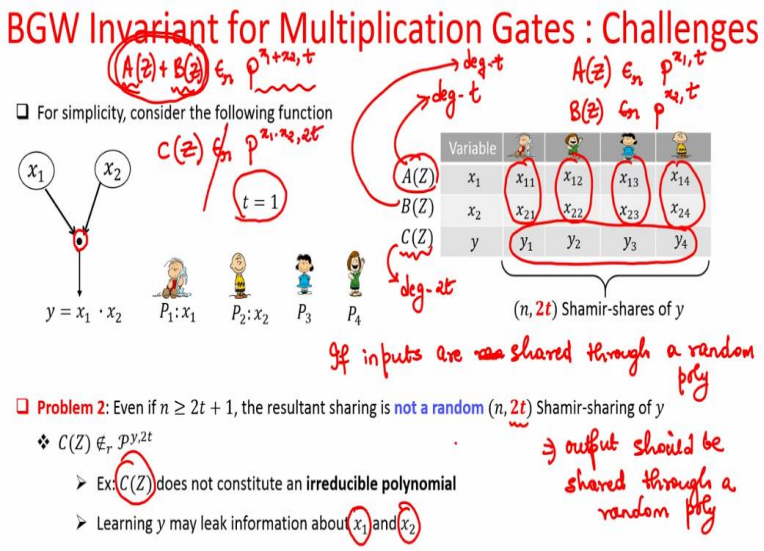
This is because, if we consider the C polynomial, it does not constitute an irreducible polynomial. What is an irreducible polynomial? So, we say a polynomial is reducible; informally we say a polynomial is reducible if it can be factorised into non-trivial polynomials. And what does non-trivial polynomials mean here? Basically, they mean polynomials whose degree is less than the polynomial which we want to factorise.

So, that is a very high level definition of a reducible polynomial. So, in this case, the C polynomial can be indeed factorised into 2 factors, namely, A polynomial and B polynomial. And the degree of both the A polynomial as well as B polynomial is strictly less than the degree of the C polynomial, because the degree of the C polynomial is $2t$ and the degree of the A polynomial is t and the degree of the B polynomial is t.

So, that means that this C polynomial definitely is not a randomly chosen polynomial from this set, because, if it would have been a randomly chosen polynomial, then, with equal probability, it could be a reducible polynomial of degree-$2t$, whose constant term would have been y, or with equal probability, it could have been an irreducible polynomial whose constant term would have been y, and degree being $2t$.

But we know in this case that, if parties just locally multiply their respective shares of x_1 and x_2, and then publicly reconstruct the C polynomial, this C polynomial is not one of the polynomials randomly chosen from this set $\mathcal{P}^{(y,2t)}$; because we know definitely for sure that it is definitely a reducible polynomial, because the A polynomial and the B polynomial constitute the factors of this C polynomial. That means, this C polynomial is no longer a random polynomial.

(Refer Slide Time: 26:11)



And this itself could be a privacy breach, because learning this C polynomial might end up leaking something about x_1 and x_2. And the problem that is happening here is, it is not only that the degree of sharing has increased, it has become $2t$, the BGW invariant says that if the

inputs are shared through a random polynomial, then the output should be shared through a random polynomial.

That is the invariant. And only when this invariant is maintained, we can argue the security; because, then we can argue that for each intermediate value, adversary just sees t random shares lying on a random polynomial whose constant term would have been that intermediate value. But t random shares could be random shares on any polynomial; hence, the adversary cannot infer anything.

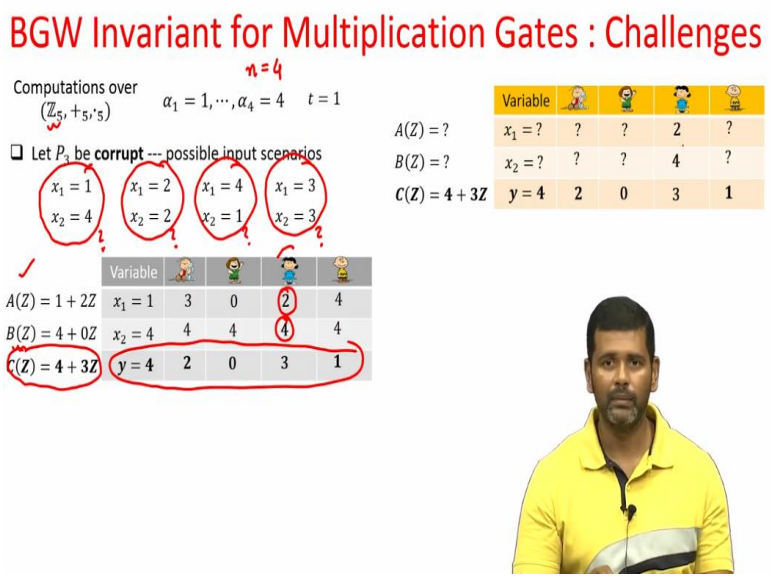
And remember, for the linear gates, if A polynomial is a randomly chosen polynomial whose degree is t; that means, for the case of linear gate, if $A(Z)$ is the member of a set of polynomials, it is a random member of the set of all polynomials of degree-t whose constant term is x_1. B is the random member of the set of all possible polynomials of degree-t, whose constant term is x_2.

And then, if parties just locally add their respective shares of A and B, the resultant C polynomial, namely the $A(Z) + B(Z)$ polynomial, it also constitutes a random member of the set of all possible polynomials of degree-t whose constant term is$x_1 + x_2$. Because, if the coefficients of A are randomly chosen; of course, except the constant term; and if same is the case for the B polynomial, then, if you take component wise the coefficients and add them, that will give you the coefficient of Z^i for this sum polynomial.

And if the coefficients of Z^i in the A polynomial and the B polynomial were random, you add 2 random coefficients; that will give you a random coefficient for Z^i in the C polynomial. So, that is why this sum polynomial is still a random member of this set of all possible polynomials of degree-t, whose constant term would have been$x_1 + x_2$; but that is not happening for the case of multiplication.

Even though the A and B polynomials respectively are random members from the respective set of polynomials, I cannot say that this C polynomial is a random member from the set of all possible polynomials of degree-$2t$ whose constant term is $x_1 \cdot x_2$; I cannot make this claim. And that will lead to a security breach, privacy breach.

(Refer Slide Time: 29:32)



So, let me demonstrate that concretely. So, again, I take the setting where n = 4, all computations performed over this field. Say x_1 is 1, and is shared through a 1-degree polynomial. This is the A polynomial $1 + 2Z$ and these are the vector of shares for x_1. The value x_2 is 4 shared through the polynomial $4 + 0Z$. And this is the vector of shares for x_2. And I consider a scenario where P_3 is corrupt.

So, that is why all the values in this table which are in bold are the values which will be seen by the corrupt P_3 using which it will try to now analyse and learn something additional about x_1 and x_2. So, P_3 is the corrupt party. This will be the view of corrupt P_3. It will see the share of x_1; it will see the share of x_2; and it will see all the shares of y, because they will be made public; and she will be seeing the C polynomial.

So, correctness is not an issue here, because degree of C polynomial will be 2. So, you might be wondering that it is degree-1 here; no; in general, it could be degree-2; because, A polynomial could have degree-1, B polynomial could have degree-1, if you multiply, you might have a term of the form Z^2 in the C polynomial; but in this case, the coefficient of Z^2 is turning out to be 0, but overall, C polynomial could be of degree-2.

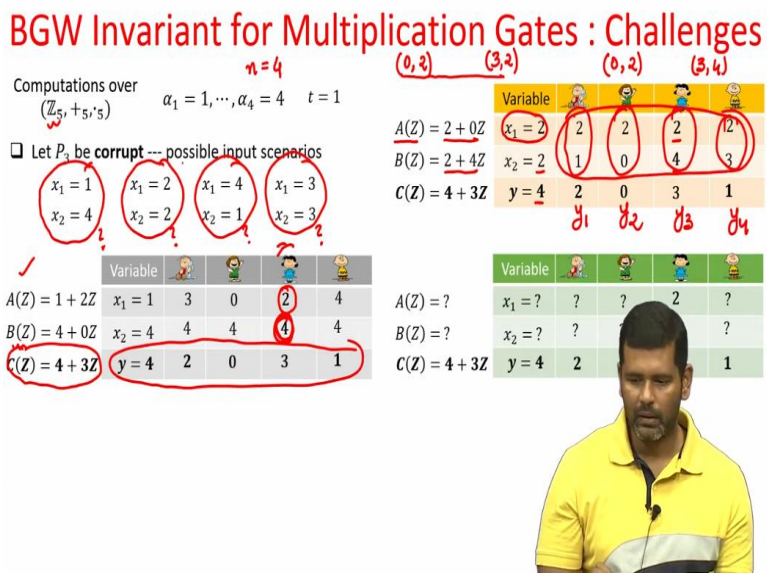
And we have sufficient number of parties to reconstruct back this Z polynomial. So, the correctness is not an issue in this particular example. Now, what the corrupt P_3 might do once the protocol is over? She is learning the final outcome to be 4; so, from her viewpoint, this could be one possible scenario with which the protocol has been executed. That means, she

might be analysing in her mind that it could be the case that input x_1 was 1 and input x_2 was 4, and the values that I have seen actually corresponds to that case.

Or, from her viewpoint, it could be the case that x_1 = 2 and x_2 = 2; or this could be the input scenario or this could be the input scenario. If at all this protocol would have been satisfying the privacy property, then adversary or the corrupt P_3 could not pinpoint whether she has seen the execution with respect to this input scenario or this input scenario or this input scenario or this input scenario.

And remember, for the case of linear functions, we have seen, we have rigorously proved that indeed, adding the shares of input values and then publicly reconstruct the function output does not reveal anything additional about the inputs of the parties other than what can be inferred from the function output. But in this particular case where the function is the multiplication of inputs, we will see that, that is not the case. So, the question marks in the table are the unknown value from the viewpoint of the corrupt P_3. So, now, imagine the corrupt P_3 makes a hypothesis that, can it be possible that x_1 was 2?

(Refer Slide Time: 32:50)



As soon as she fixes x_1 = 2; and remember the unknown A polynomial was a 1-degree polynomial. So, now, as soon as she fixes x_1 = 2, that means, she is fixing the A polynomial passing through the point (0, 2). And she has seen the share 2 during the protocol. That means, (3, 2) also is a point on that unknown A polynomial.

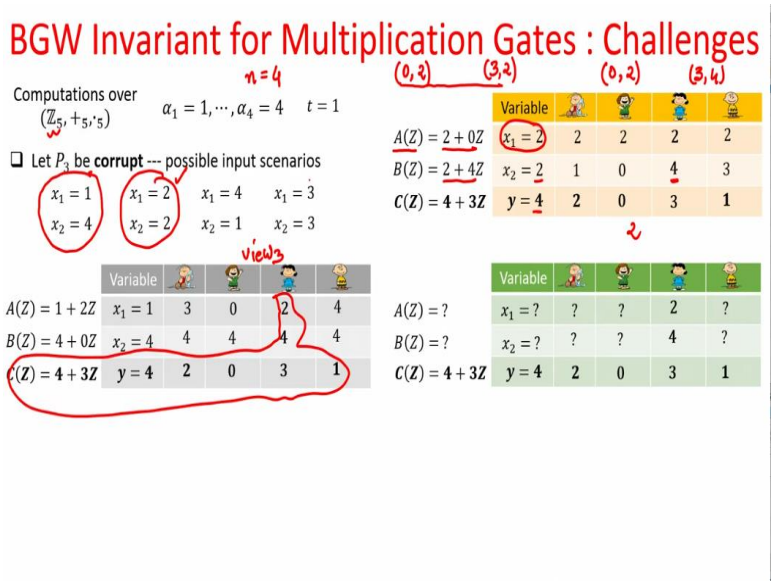
And now, once 2 points are fixed, that automatically fixes the A polynomial. So, that fixes the A polynomial from the viewpoint of corrupt P_3. And if this would have been the A polynomial, then the shares of P_1, P_2 and P_4 would have taken these values. Now, once P_3 has fixed $x_1 = 2$ in her mind, and her final result is 4, that automatically fixes x_2 being 2. And now, if x_2 is fixed to be 2, that means, adversary or the corrupt P_3 is now considering the case where the B polynomial should have passed through the points (0, 2) and (3, 4).

Why (3, 4)? Because her share for the unknown B polynomial that she has seen is actually 4. So, fixing $x_2 = 2$ fixes the B polynomial to $2 + 4Z$. And if the B polynomial would have been $2 + 4Z$, then these unknown shares of P_1, P_2 and P_4 for x_2 would have been this. And now it matches with whatever adversary or the corrupt P_3 had actually seen during the protocol execution.

Indeed, if x_1 was 2 and secret-shared through the polynomial $2 + 0Z$, and if x_2 was 2 and secret-shared through the 1-degree polynomial $2 + 4Z$, this will be the vector of shares for the respective parties. And after multiplying, they would have obtained this y_1, this y_2 indeed. They would constitute the corresponding y values and they would have made these y values public.

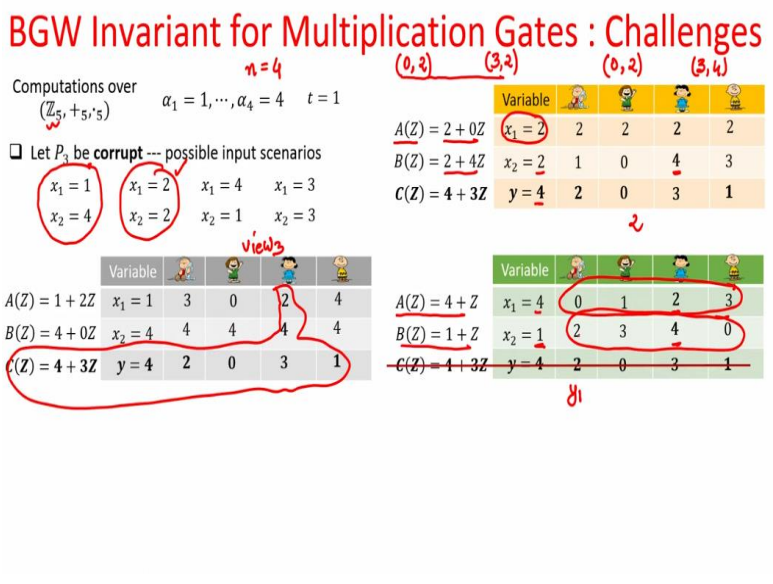
And hence, it is quite possible that adversary, the corrupt P_3 had actually seen an execution where x_1 was 2 and x_2 was 2. That is quite possible. So, this cannot be ruled out. Now, adversary takes the same set of values which she had seen in the protocol, namely her view. So, basically, the values in the bold is the view of the corrupt P_3. And we are trying to argue here whether we can say that view 3 is consistent with all possible x_1, x_2 whose product is 4?

(Refer Slide Time: 35:56)



So, we had shown that the view 3; so, this is view 3, the values in bold here; and we had shown that it is consistent with $x_1 = 2$ and $x_2 = 2$. Of course, it is also consistent with $x_1 = 1$ and $x_2 = 4$, because that precisely is the values with which the protocol has been executed. Now, the remaining 2 configurations are left from the viewpoint of the current P_3.

(Refer Slide Time: 36:29)

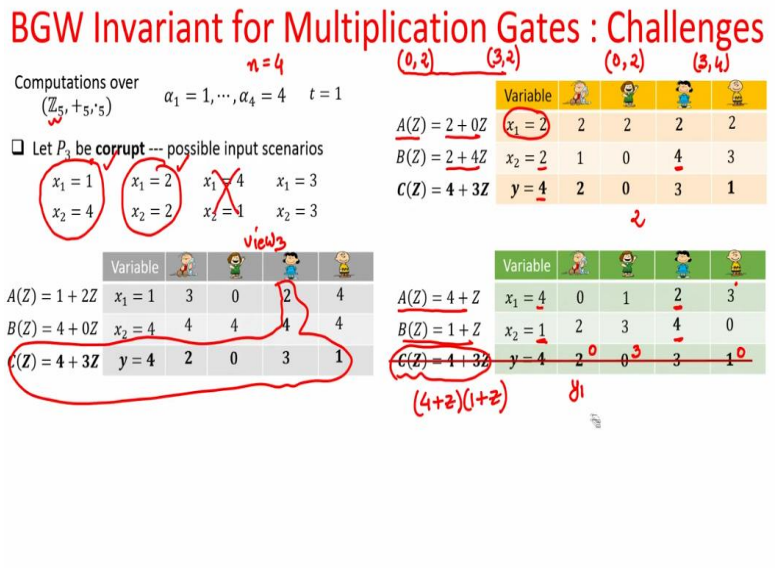


So, let us consider the case where P_3 is fixing x_1 to be 4 and x_2 to be 1. And she is asking in her mind, analysing in her mind, can it be the case that my view has resulted because I have participated in an execution where x_1 is 4 and x_2 is 1? So, as soon as x_1 is fixed to 4 and x_2 is fixed to 1, then based on the argument that we had seen for the previous input configuration, it fixes the A polynomial to this value and B polynomial to this value.

That means, if x_1 would have been 4 and if P_3's share would have been 2, then, this would have been the full vector of x_1 shares. In the same way, if x_2 would have been 1 and if P_3's share would have been 4, then this would have been the full vector of x_2 shares. If this would have been the case, then it cannot be possible that P_1, after multiplying its shares of x_1 and x_2, would have obtained $y_1 = 2$.

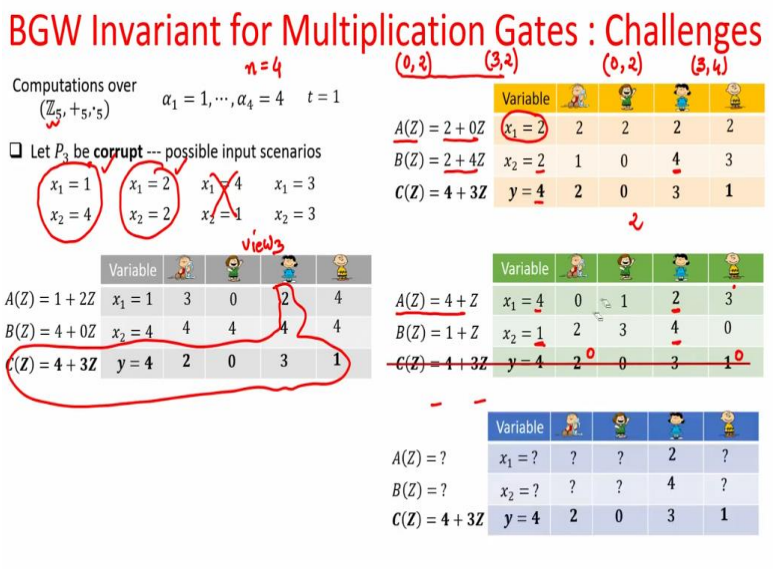
No, that is not possible; because, his share of x_1 would have been 0, his share of x_2 would have been 2, and 0 into 2 should have given 0. And that means, if this configuration or this execution has been done, then P_1 should have broadcasted 0 instead of 2 as its share.

(Refer Slide Time: 37:57)

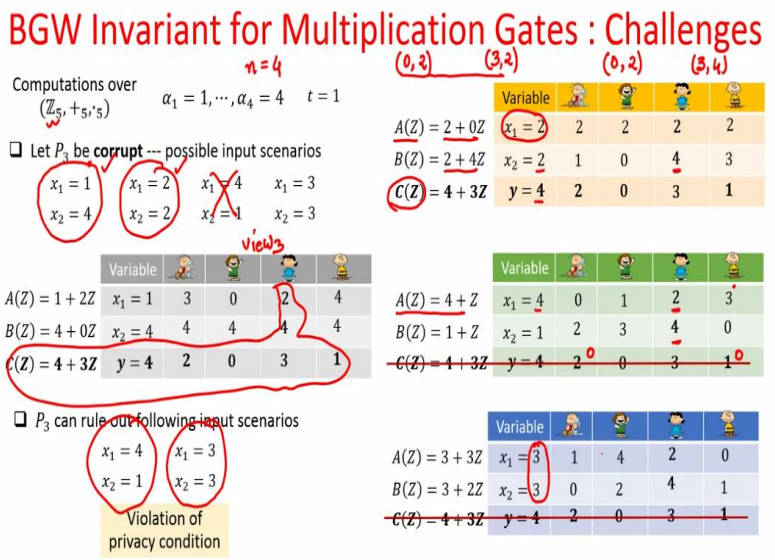


In the same way, P_2 should have made public 3 as its share of y, instead of 0. And P_4 should have made public 0 as its share of y, instead of 1. And this should not have been the C polynomial. C polynomial should have been the product of the polynomials $4 + Z$ and $1 + Z$, which is not the case. That means, now, corrupt P_3 can say that, definitely I have not seen an execution with respect to x_1 = 4 and x_2 = 1. So, that is ruled out from her viewpoint.

(Refer Slide Time: 38:35)



(Refer Slide Time: 38:37)



And in the same way, we can show here that she can rule out that view 3 is not consistent, even with $x_1 = 3$ and $x_2 = 3$; that is not possible. That means, P_3 can rule out that, definitely these are not the pair of inputs of P_1 and P_2. And that is a violation of the privacy condition. And again, I stress, why this issue is happening? This issue is happening because this C polynomial is not a random polynomial from the set of all possible polynomials of degree-$2t$ whose constant term is $x_1 \cdot x_2$; and that is the problem here.

(Refer Slide Time: 39:20)

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
  - Dario Catalano, Ronald Cramer, Ivan Bjerre Damgård, Giovanni Di Crescenzo, David Pointcheval: Contemporary cryptology. Advanced courses in mathematics : CRM Barcelona, Birkhäuser 2005, ISBN 978-3-7643-7294-1, pp. I-VIII, 1-237

So, with that I conclude this lecture. And to summarise, in this lecture, we have seen that there are 2 challenges associated with evaluating the multiplication gates. The first challenge is associated related to correctness itself. Namely, if we just asked the parties to locally multiply their respective shares of the inputs of a multiplication gate, then the degree of sharing blows up, and we will not have sufficient number of parties to reconstruct back that secret-shared value.

And the second problem is that, even if we have sufficient number of parties to reconstruct back that secret-shared output whose degree has become twice, the resultant vector of shares does not lie on a random polynomial of degree-$2t$. For instance, it is not an irreducible polynomial; and that might itself be an issue of privacy. Thank you.