

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

Module - 4
Lecture - 21
The BGW MPC Protocol for Linear Functions: Security Analysis

Hello everyone. Welcome to this lecture. So, in this lecture, we will continue our discussion on the BGW MPC protocol for securely computing linear functions, and we will perform its security analysis.

(Refer Slide Time: 00:41)

Lecture Overview


- The BGW protocol for linear functions
 - ❖ Security analysis

(Refer Slide Time: 00:42)

BGW Protocol for Linear Functions: Example

$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 2$ $c_1 = c_2 = c_3 = c_4 = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ $y = 4$

$x_1 = 2$ $x_4 = 0$



$x_2 = 1$ 

$x_3 = 1$ 

So, before going into the security analysis formally, let us try to understand the security through an example itself, because that will help us to understand the intuition regarding why the protocol is secure. And I am considering here the toy example, where my number of parties is 4 and the threshold is 2. That means, any 2 out of the 4 parties can be passively corrupt. And I am performing all the computations over this field.

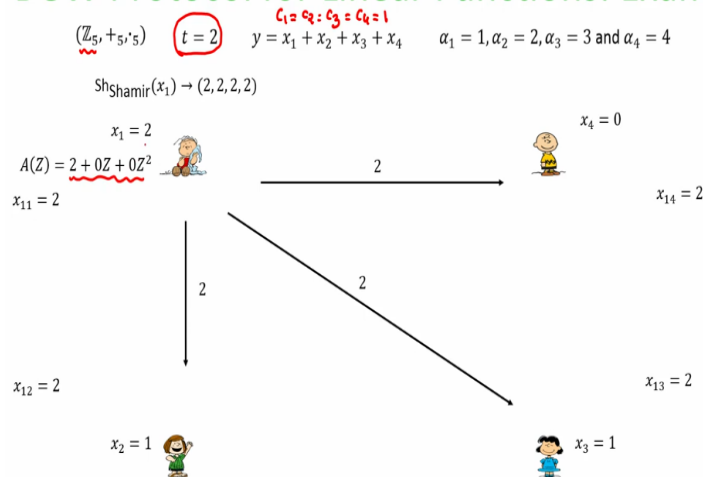
And my plus and dot operations are $+\mathbb{F}_5$ and $\cdot\mathbb{F}_5$ respectively. And I am taking the case where my constant c_1, c_2, c_3 and c_4 are all 1. Namely, I am taking simply the sum function, and I am setting my evaluation points to 1, 2, 3 and 4 respectively. Well, anyhow, these are the only non-zero evaluation points available for this specific field. But in general, if I take a finite field, it could be any non-zero elements which are distinct, and they can be chosen as your evaluation points.

And I am taking an execution scenario where the inputs are 2, 1, 1 and 0. Namely, the final sum will be 4. So, that means, finally the parties will learn the sum $y = 4$; and depending upon which 2 parties are corrupt, they will learn anyhow, what could be the overall sum of the remaining 2 parties. That much information is allowed to be learnt from the inputs and output of the correct parties; but we have to see that, apart from that, whatever messages the parties exchange in this BGW MPC protocol, that does not help those 2 bad parties to learn anything additional.

So, for instance, if I say the first 2 parties are under the control of adversary, they will anyhow learn that the inputs of the remaining 2 parties, together it is 2; because the final sum is 4. But whether it is 1, 1 or whether it is 0, 2 or whether it is 2, 0, that much information should not be revealed, because that is not allowed to be learnt to these corrupt parties as per the security definition. That is what we have to ensure.

(Refer Slide Time: 03:19)

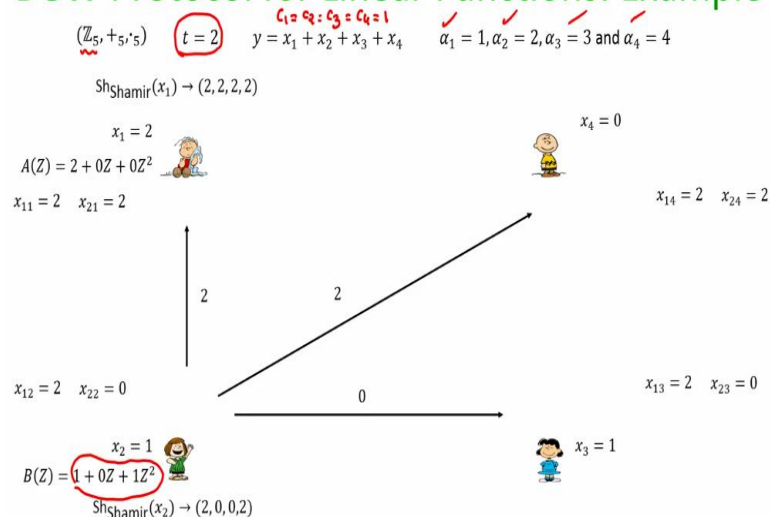
BGW Protocol for Linear Functions: Example



So, to start with, P_1 will secret-share its input. So, suppose this is its sharing polynomial; as I said, this will be considered as a 2-degree polynomial. So, all the sharing polynomials have to be degree-2. So, this is a 2-degree polynomial, even though the coefficients of Z and Z^2 are 0; but overall, it will be considered as a 2-degree polynomial. And the shares that P_1 will compute with respect to this sharing polynomial will be 2, 2, 2 and 2, which will be communicated over the private channels to the respective parties; and that will be the individual shares of the input of the first party.

(Refer Slide Time: 04:13)

BGW Protocol for Linear Functions: Example

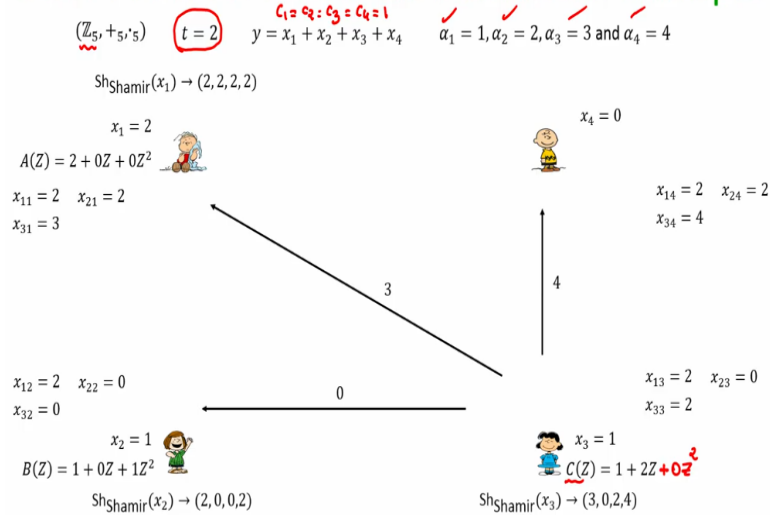


Now, suppose the second party decides to secret-share its input 1 through this sharing polynomial; call this sharing polynomial is B polynomial. Again, it could be any B polynomial whose constant term is 1; and the remaining 2 coefficients can be randomly chosen from \mathbb{Z}_5 . And now, with respect to this sharing polynomial, this B polynomial evaluated that is

$\alpha_1, \alpha_2, \alpha_3, \alpha_4$ will produce the shares 2, 0, 0, 2. And the shares will be communicated over the private channel. So, that will give the individual parties their respective shares of x_2 .

(Refer Slide Time: 04:55)

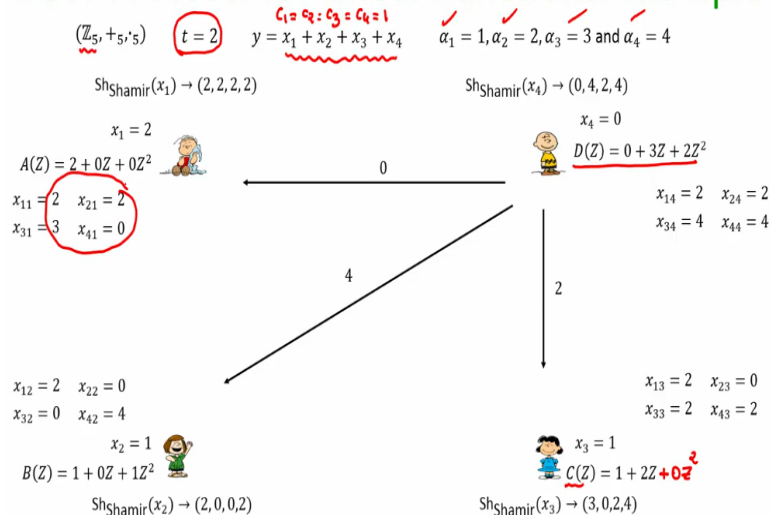
BGW Protocol for Linear Functions: Example



Suppose, the third party decides to secret-share its input through this polynomial; so, I have not written down $0 \cdot Z^2$; so, it is a polynomial of degree-2. And with respect to this C polynomial, the shares computed by P_3 will be 3, 0, 2, 4, which it will securely communicate. And that will give the parties their respective shares for the input x_3 .

(Refer Slide Time: 05:19)

BGW Protocol for Linear Functions: Example



And say, for the fourth party, it decides to secret-share its input through this random D polynomial; compute the shares and distribute to the respective parties. That is the input stage. And after that, what the parties have to do? They have to just add their respective shares of

x_1, x_2, x_3 and x_4 . That means, P_1 has to add these 4 shares, and then add as per addition modulo 5. So, $3 + 2$ is 5; $5 + 2$ is 7; and then, 7 modulo 5 will be 2.

(Refer Slide Time: 05:53)

BGW Protocol for Linear Functions: Example

$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 2$ $c_1 = c_2 = c_3 = c_4 = 1$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$

$\text{ShShamir}(x_1) \rightarrow (2, 2, 2, 2)$
 $x_1 = 2$
 $A(Z) = 2 + 0Z + 0Z^2$
 $x_{11} = 2$ $x_{21} = 2$
 $x_{31} = 3$ $x_{41} = 0$ $y_1 = 2$

$\text{ShShamir}(x_4) \rightarrow (0, 4, 2, 4)$
 $x_4 = 0$
 $D(Z) = 0 + 3Z + 2Z^2$
 $x_{14} = 2$ $x_{24} = 2$
 $x_{34} = 4$ $x_{44} = 4$

$x_{12} = 2$ $x_{22} = 0$
 $x_{32} = 0$ $x_{42} = 4$
 $x_2 = 1$
 $B(Z) = 1 + 0Z + 1Z^2$
 $\text{ShShamir}(x_2) \rightarrow (2, 0, 0, 2)$

$x_{13} = 2$ $x_{23} = 0$
 $x_{33} = 2$ $x_{43} = 2$
 $x_3 = 1$
 $C(Z) = 1 + 2Z + 0Z^2$
 $\text{ShShamir}(x_3) \rightarrow (3, 0, 2, 4)$

That will be party P_1 's share for the final output y . Similarly, P_2 has to go and add it is all shares of x_1, x_2, x_3, x_4 . So, $4 + 2$ is 6; and 6 modulo 5 will be 1. So, by the way, in the animation, you compute your final share of y and communicate that to everyone.

(Refer Slide Time: 06:19)

BGW Protocol for Linear Functions: Example

$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 2$ $c_1 = c_2 = c_3 = c_4 = 1$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$

$\text{ShShamir}(x_1) \rightarrow (2, 2, 2, 2)$
 $x_1 = 2$
 $A(Z) = 2 + 0Z + 0Z^2$
 $x_{11} = 2$ $x_{21} = 2$
 $x_{31} = 3$ $x_{41} = 0$ $y_1 = 2$

$\text{ShShamir}(x_4) \rightarrow (0, 4, 2, 4)$
 $x_4 = 0$
 $D(Z) = 0 + 3Z + 2Z^2$
 $x_{14} = 2$ $x_{24} = 2$
 $x_{34} = 4$ $x_{44} = 4$

$x_{12} = 2$ $x_{22} = 0$
 $x_{32} = 0$ $x_{42} = 4$
 $x_2 = 1$
 $B(Z) = 1 + 0Z + 1Z^2$
 $\text{ShShamir}(x_2) \rightarrow (2, 0, 0, 2)$

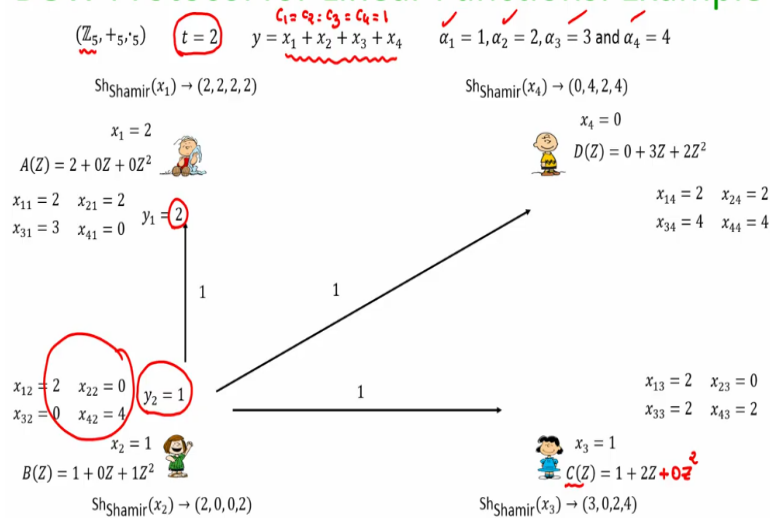
$x_{13} = 2$ $x_{23} = 0$
 $x_{33} = 2$ $x_{43} = 2$
 $x_3 = 1$
 $C(Z) = 1 + 2Z + 0Z^2$
 $\text{ShShamir}(x_3) \rightarrow (3, 0, 2, 4)$

Arrows from $y_1 = 2$ to other nodes are labeled with '2'.

So, y_1 will be communicated to everyone.

(Refer Slide Time: 06:23)

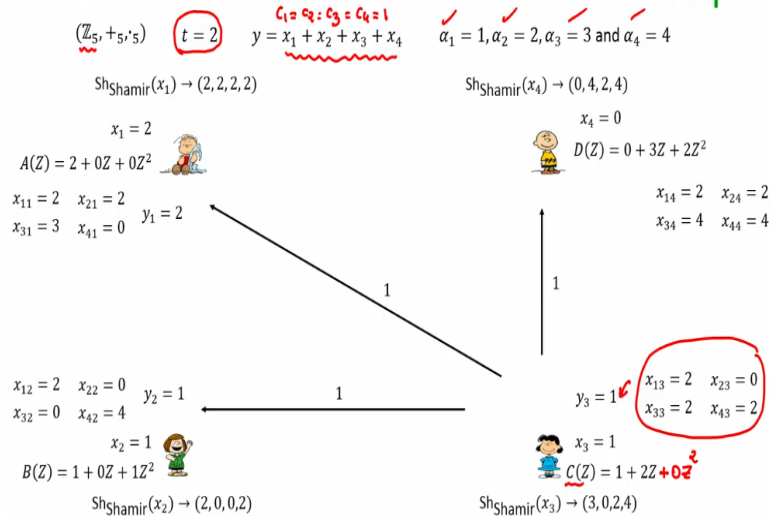
BGW Protocol for Linear Functions: Example



Similarly, the share of y for P_2 will be 1, because $4 + 2$ will be 6; 6 modulo 5 is 1. And then, it will communicate to everyone. But remember, in the actual protocol, this is not done like this. Every party obtains its respective share of y and simultaneously communicates to everyone; but for the animation, I am showing it in a sequential way, remember.

(Refer Slide Time: 06:52)

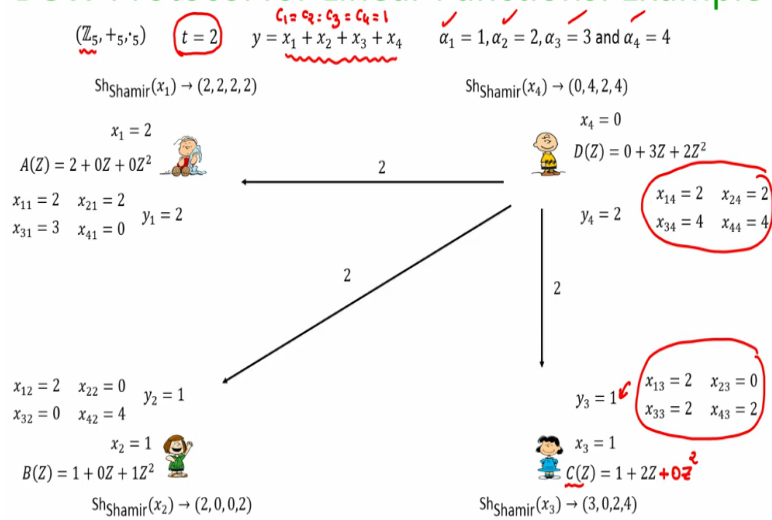
BGW Protocol for Linear Functions: Example



Now, P_3 will do the following: If it adds its shares of x_1, x_2, x_3, x_4 , that will be 6; 6 modulo of 5 will be 1. That is share of y it will communicate to everyone.

(Refer Slide Time: 07:04)

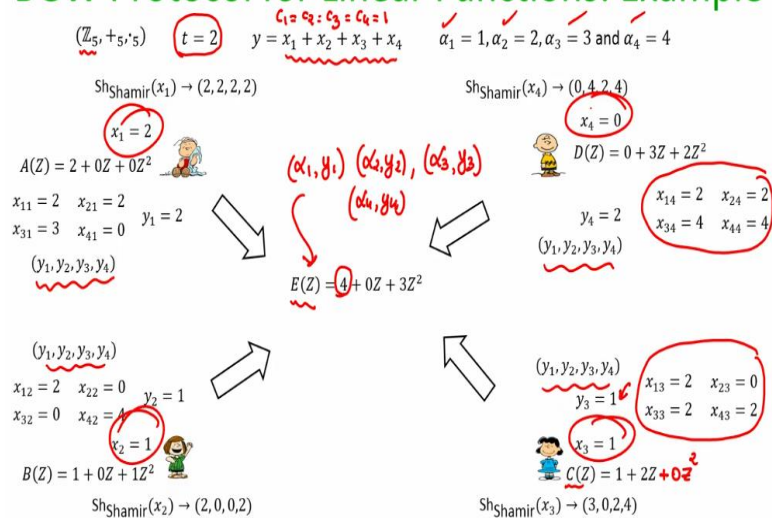
BGW Protocol for Linear Functions: Example



And then, P_4 finally goes and add its shares of x_1, x_2, x_3 and x_4 . That will be 12; 12 modulo 5 will be 2. And now, it will communicate that to everyone.

(Refer Slide Time: 07:16)

BGW Protocol for Linear Functions: Example



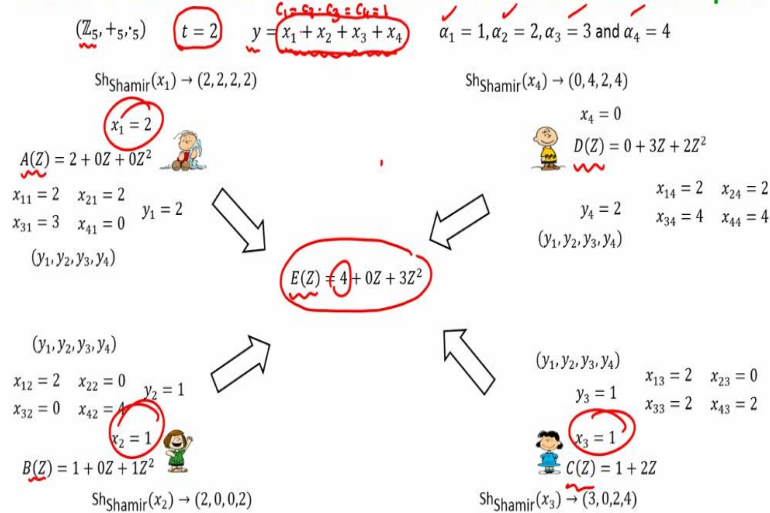
So, now, everyone will have the full vector of shares of y_1, y_2, y_3, y_4 . And now, they have to interpolate these and get back the value of y . So, interpolate in the sense, they have to now interpolate $(\alpha_1, y_1); (\alpha_2, y_2); (\alpha_3, y_3);$ and (α_4, y_4) ; they have to interpolate. And now, if they interpolate these values, they obtain this E polynomial. And the constant term of this E polynomial will be taken as the final output.

That is the execution of BGW protocol, assuming $x_1 = 2, x_2 = 1, x_3 = 1, x_4 = 0$. I stress, again if the BGW protocol is executed by the parties with the *same* set of inputs x_1, x_2, x_3, x_4 ,

then it *need not* be the case that they pick the same A polynomials, B polynomials, C polynomials and D polynomials for sharing, and get the same E polynomial.

(Refer Slide Time: 08:29)

BGW Protocol for Linear Functions: Example



It could be any random A polynomial, any random B polynomial, any random C polynomial, any random D polynomial, of course, with their respective constant terms being 2 and 1 and 1 and 0. And finally, they will obtain; an E polynomial which could be another random polynomial of degree-2 except that its constant term will be 4; because, there is an internal randomness used as part of the BGW MPC protocol, even though your function y is a deterministic output of x_1, x_2, x_3, x_4 .

(Refer Slide Time: 09:09)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt. Ady's view in bold

Variable	Value	P_1	P_2	P_3	P_4
x_1	2	2	2	2	2
x_2	1	2	0	0	2
x_3	1	3	0	2	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$\rightarrow A(z)$
 $\rightarrow B(z)?$
 $\rightarrow C(z)$
 $\rightarrow D(z)$
 $\rightarrow E(z)$



So, now, we will try to understand the privacy through this example, which will help us to understand the intuition. We take the case where, say the first and the fourth party, they are

corrupt. And basically, I have constructed here a table, where basically I have written down all the variables and the values that they have taken during the concrete execution. And along the first row, the values that are there, basically they are the shares of the value x_1 that the respective parties, P_1, P_2, P_3 and P_4 would have seen.

Along the second row, I have denoted the shares. So, basically, the first row values, they are the values on $A(Z)$ polynomial, which party P_1 has taken and during the execution. The second row basically are the shares computed as per the B polynomial. The third row denotes the C polynomial; the fourth row denotes the D polynomial; and the last row denotes the E polynomial which parties finally obtain by interpolating the shares y_1, y_2, y_3, y_4 .

And in the table, you see that certain values are highlighted, they are bold. So, for instance, the first row is completely bold, because I am considering the case where P_1 and P_4 , I am assuming to be corrupt. So, that means, in this table, the values P_1 and P_4 see in the protocol is their view, and that is highlighted in the bold. So, since P_1 is under the control of adversary, it will completely know the polynomial $A(Z)$.

And hence, it will know that what are the shares P_1 has distributed. That is why, the values along the first row, they are highlighted in bold. But P_2 is not under the control of the adversary, so, that is why, the value of x_2 is not highlighted in bold. But for x_2 , there are 2 shares which adversary learns, namely, the share which is given to party 1 and a share which is given to party 4.

So, that is why, this value and this value in the second row, they are highlighted in bold. Similarly, x_3 is not known to the adversary. But for x_3 , P_1 and P_4 , they give their respective shares to the adversary. So, that is why, the first value here, first share here and the fourth share, they are highlighted in bold. And x_4 is now completely known to adversary. And the $D(Z)$ polynomial is completely known to the adversary, because, adversary I am assuming is controlling party P_4 .

So, that is why, it will know the complete fourth row. And now, the last row, all the values are public, because y_1 is made public, y_2 is made public, y_3 is made public, y_4 is made public. And since the polynomial $E(Z)$ is interpolated in public, the value y is known. So, that means,

from the viewpoint of the adversary, the values which are not highlighted in bold are the unknown values.

(Refer Slide Time: 12:31)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt Adv's view in bold

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	0	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$x_1 = 2, x_3 = 0$

$x_2 = 1, x_3 = 1$

$x_2 = 0, x_3 = 2$

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

So, let me put them in question mark, from the viewpoint of P_1 and P_4 . Whereas, all other values which are not with question mark, they are actually known to the adversary, where I am assuming, I stress, I assuming that P_1 and P_4 are corrupt. If it would have been another set of 2 corrupt parties, then the question marks will be changed here. That means, that question marks will be in some other entries and so on.

So, now, what adversary will try to do here, assuming P_1 and P_4 are bad? They will take this table; and of course they know that x_1 is 2, x_4 is 0 and the final sum is 4. They will try to figure out what could be these 2 question marks, because that is a private information. So, now, adversary might try to analyse and ask many questions in its mind. It may ask in its mind, is it the case that the values that I have seen in the protocol are actually corresponding to the case where P_2 's input was 0 and P_3 's input was 2?

That is one possibility for the adversary. Or, adversary could ask in its mind that, is this table of information some of which is known, some of which is not known could occur for the case where x_2 is 2 and x_3 is 0. Or, is it the case that x_2 was 1 and x_3 was 1? That means, these are the various possibilities from adversary's point of view, based on this information that he has seen.

Now, it at all our BGW MPC protocol is private; private in the sense, if it ensures the privacy property, then, even if adversary does this analysis, and even if its computing power is unbounded, he should not be able to figure out whether actually these values which he has seen here are with respect to the input configuration being 0, 2, or input configuration with respect to 2, 0, or with respect to input configuration 1, 1.

If that is the case, then basically, we end up showing that adversary's view here is consistent with every candidate x_2, x_3 which along with x_1 being 2 and x_4 being 0, sum up to the value 4. And we will show that actually this is indeed the case. So, let us see.

(Refer Slide Time: 15:08)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Adv's view in bold

Possible, if $x_2 = 0$, $B(Z) = 2Z^2$ and $x_3 = 2$, $C(Z) = 4Z^2 + 2Z + 2$

Variable	Value				
x_1	2	2	2	2	2
x_2	0	2	3	3	2
x_3	2	3	2	4	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$x_2 = 0, x_3 = 2$

Now, the adversary is thinking in its mind that, can it be the case that these question marks here, this x_2 being a question mark and x_3 being a question mark, is actually 0 and 2 respectively? But, you see, he is not changing the shares that he has seen, namely, the first share of the question mark here, first share of x_2 and the fourth share of x_2 , because, they are actually fixed.

That is what adversary has seen here. It is just asking in his mind that, can it be the case that x_2 was 0, its first share was 2 and the fourth share was 2? Well, that is quite possible if P_2 would have selected this polynomial for secret-sharing, namely $B(Z) = 2Z^2$. But adversary does not know what is the value of the B polynomial which P_2 would have selected for secret-sharing his unknown x_2 . For that unknown x_2 , he has just got 2 shares.

Those 2 shares could be with respect to any B polynomial whose constant term could be any x_2 ; that is what is the privacy property of secret-sharing. So, that means, if x_2 is fixed to 0, that, along with the first share being 2 and the fourth share being 2, automatically fixes the B polynomial which P_2 would have picked; because, they are now total 3 points, and 3 points uniquely determine a 2-degree polynomial.

(Refer Slide Time: 16:46)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt Adv's view in bold

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$x_2 = 0, x_3 = 2$

Variable	Value				
x_1	2	2	2	2	2
x_2	2				2
x_3	?	3			4
x_4	0	0	4	2	4
y	4	2	1	1	2

In the same way, adversary does not know what is this question mark in place of x_3 's value. He only knows that, okay, there was some 2-degree polynomial with some unknown constant, but that polynomial evaluated at α_1 should have produced 3, and evaluated at α_4 should have produced 4. What could be that unknown polynomial? Well, that unknown polynomial could also be this C polynomial whose constant term would have been 2.

(Refer Slide Time: 17:08)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt Adv's view in bold

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$x_2 = 0, x_3 = 2$

$x_2 = 2, x_3 = 0$

Possible, if $x_2 = 0, B(Z) = 2Z^2$ and $x_3 = 2, C(Z) = 4Z^2 + 2Z + 2$

Variable	Value				
x_1	2	2	2	2	2
x_2	0	2	3	3	2
x_3	2	3	2	4	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

And now, you see, this table matches the configuration of information which adversary has actually seen while participating in the process. That means, it is quite possible that the information which adversary has seen during the execution of the BGW MPC protocol is matching as per this configuration as well. That means, it could be the case that x_2 was 0, x_3 was 0, the B polynomial was this, C polynomial was this, and it is consistent with whatever adversary has seen in the actual protocol.

That is quite possible; adversary cannot rule it out. Or, it could be the case that these unknown question marks here, corresponds to x_2 being 2 and x_3 being 0, and the remaining things fixed as per the values that adversary has seen.

(Refer Slide Time: 18:04)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt. Adv's view in bold

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Possible, if $x_2 = 0, x_3 = 2$

Variable	Value				
x_1	2	2	2	2	2
x_2	0	2	3	3	2
x_3	2	3	2	4	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Possible, if $x_2 = 2, x_3 = 0$

Variable	Value				
x_1	2	2	2	2	2
x_2	2	2	2	2	2
x_3	0	3	3	0	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Possible, if $x_2 = 2, x_3 = 1$

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Handwritten notes: $2+0Z+0Z^2$, $2+0Z+0Z^2$, $2+0Z+0Z^2$, $x_2=0, x_3=2$, $x_2=2, x_3=0$, $x_2=2, x_3=1$.

It is quite possible that this configuration also would have produced the same information which adversary has seen while participating in the protocol. It is quite possible here; you can see here. It could be the case that $B(Z)$ was shared through the polynomial $2 + 0 \cdot Z + 0 \cdot Z^2$, whose first share would have been 2 and fourth share would have been 2. And x_3 was 0 and secret-shared through $0 + 2 \cdot Z + Z^2$, producing the first share being 3 and the fourth share being 4.

And then, P_1 announces 2 as its share of y , and P_2 announces 1 as its share of y , and P_3 announces 1 as its share of y , and P_4 announces 2 as its share of y ; and all together, they interpolate back and give the value 4. That is also quite possible.

(Refer Slide Time: 18:59)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt Adv's view in bold

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$2+0Z+0Z^2$

Possible, if $x_2 = 2, B(Z) = 2$ and $x_3 = 0, C(Z) = 2Z + Z^2$

Variable	Value				
x_1	2	2	2	2	2
x_2	2	2	2	2	2
x_3	0	3	3	0	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$2+0Z+0Z^2$

Possible, if $x_2 = 2, B(Z) = 2$ and $x_3 = 0, C(Z) = 2Z + Z^2$

$x_2 = 2, x_3 = 0$ $x_2 = 1, x_3 = 1$

Possible, if $x_2 = 0, B(Z) = 2Z^2$ and $x_3 = 2, C(Z) = 4Z^2 + 2Z + 2$

Variable	Value				
x_1	2	2	2	2	2
x_2	0	2	3	3	2
x_3	2	3	2	4	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Possible, if $x_2 = 1, B(Z) = 1 + 0Z + 1Z^2$ and $x_3 = 1, C(Z) = 1 + 2Z$

Variable	Value				
x_1	2	2	2	2	2
x_2	1	2	0	0	2
x_3	1	3	0	2	4
x_4	0	0	4	2	4
y	4	2	1	1	2

And in the same way, it is quite possible that the unknown x_2, x_3 could be 1, 1 as well, shared through this respective B and C polynomial, and matching with everything that adversary has actually seen while participating in the protocol. So, what I have demonstrated in this example is the following: That even though adversary has seen the values which are highlighted in bold in this table of information; so, this is the entire view of adversary.

(Refer Slide Time: 19:29)

BGW Protocol for Linear Functions: Privacy

Let P_1, P_4 be corrupt Adv's view in bold

Variable	Value				
x_1	2	2	2	2	2
x_2	?	2	?	?	2
x_3	?	3	?	?	4
x_4	0	0	4	2	4
y	4	2	1	1	2

$x_2 + x_3 = 2$

Possible, if $x_2 = 2, B(Z) = 2$ and $x_3 = 0, C(Z) = 2Z + Z^2$

Variable	Value				
x_1	2	2	2	2	2
x_2	2	2	2	2	2
x_3	0	3	3	0	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Possible, if $x_2 = 2, B(Z) = 2$ and $x_3 = 0, C(Z) = 2Z + Z^2$

$x_2 = 2, x_3 = 0$ $x_2 = 1, x_3 = 1$

Possible, if $x_2 = 0, B(Z) = 2Z^2$ and $x_3 = 2, C(Z) = 4Z^2 + 2Z + 2$

Variable	Value				
x_1	2	2	2	2	2
x_2	0	2	3	3	2
x_3	2	3	2	4	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Possible, if $x_2 = 1, B(Z) = 1 + 0Z + 1Z^2$ and $x_3 = 1, C(Z) = 1 + 2Z$

Variable	Value				
x_1	2	2	2	2	2
x_2	1	2	0	0	2
x_3	1	3	0	2	4
x_4	0	0	4	2	4
y	4	2	1	1	2

Adv's view consistent with every candidate x_2, x_3 such that $x_1 + x_2 + x_3 + x_4 = 4$

This is, call it as $view_{1,4}$; that is the view of adversary, because these are the values messages which it has communicated, which it received during the protocol execution. What I have shown through the example here is the following: That this view which adversary has seen could be possible for the input configuration $x_2 = 0, x_3 = 2$; or it could occur with $x_2 = 1, x_3 = 1$; or it could also occur with $x_2 = 2, x_3 = 0$, with equal probability.

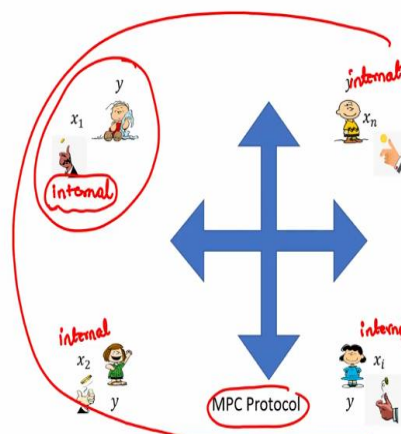
Adversary cannot say that this was the case or this was the case or this was the case; no. And hence, the information which adversary has learnt here does not help him to learn anything additional about x_2 and x_3 ; I stress, additional. Of course, it learns that $x_2 + x_3 = 2$. It learns that information, fine. This is revealed; but this is allowed to be revealed; because, again and again I stress, we want to ensure that adversary should not learn anything additional beyond what it can learn from its own input and function output.

So, that means, this example at least demonstrates that adversary does not learn any information during the BGW MPC protocol for evaluating linear functions. That means, we have shown that adversary's view is consistent with every candidate x_2, x_3 from the field, which along with $x_1 = 2$ and $x_4 = 0$, sums up to 4.

(Refer Slide Time: 21:20)

Generic MPC Protocol : Privacy Definition

view: "Everything" that P_i has during the protocol --- input, output, random coins, messages sent and received



So, now, we have to argue that why it is happening. Is it only for this specific example, because we have performed all the computations over \mathbb{Z}_5 ? No. The answer is, it is indeed going to happen irrespective of what are the values of x_1, x_2, x_3, x_4 . And we are now going to give a formal security proof; but for that, we have to first analyse or we have to first give the formal privacy definition.

When do we say that a generic MPC protocol has the privacy property? So, any MPC protocol for computing any kind of function; it could be linear function, it could have multiplication gates, it could be any kind of function, will be abstracted as follows: So, in the protocol, every party will have its own input and there will be internal random coins. These are internal random

coins which the party might be using to decide what values it has to exchange or communicate to the other parties.

So, for instance, if I take the BGW protocol, this internal randomness basically constitutes the sharing polynomials which are picked randomly by the parties to secret-share their respective inputs. But if I consider any abstract MPC protocol, there will be some internal randomness. I do not know what kind of internal randomness are used inside the MPC protocol, because, right now, I am considering an abstract MPC protocol.

And now, once the input and the internal randomness is decided, and the steps of the MPC protocol are there fixed, the parties exchange messages, where the messages are decided based on what exactly are the inputs of the parties, what are the random values that they have generated, namely, the random coins, and what messages they have received from the other parties. And then, finally, they obtain the function output.

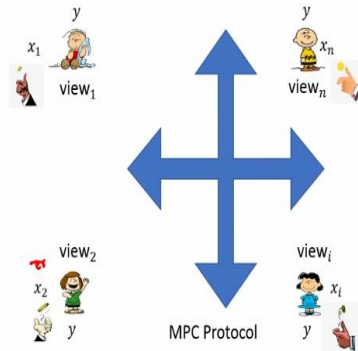
That will be the way we can abstract out any generic MPC protocol. So, now, for any generic MPC protocol, we can define $view_i$ to be the view of i^{th} party. And this is consisting of everything that P_i possesses during the execution of the protocol. So, if I consider $view_1$, say for instance; $view_1$ will have definitely the input of the party, namely x_1 ; final output of the party, namely y ; the random coins which P_1 has used to decide the messages during the protocol execution.

So, for instance, if I take the BGW protocol or the linear function, then the random coins was the random polynomial which P_1 used to compute the shares for the other parties; the messages that it has sent to the other parties. So, again, for the BGW protocol, it was the, it was the shares which it has communicated to the other parties and the final shares of y ; and the messages which P_1 would have received from the other parties during the protocol execution. Everything put together will constitute $view_1$. Similarly, I can define $view_2$ and so on.

(Refer Slide Time: 24:50)

Generic MPC Protocol : Privacy Definition

□ $view_i$: "Everything" that P_i has during the protocol --- input, output, random coins, messages sent and received
 ↓ random variable x_i, y Could vary Could vary Could vary



So, these are views $view_1, view_2, view_i, view_n$. And the important thing is that, this $view_i$ is a random variable even though it is a variable, because, the input could take any value, the output could take any value, the random coins could take any value, the messages sent and received could be any value. So, that is why this $view_i$ is a variable, but it is actually a random variable.

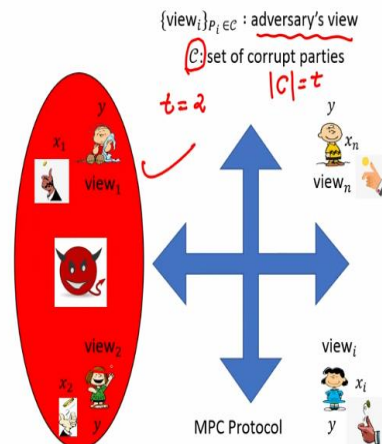
Why it is a random variable, even if I fix the function input and output? Because, the internal random coins could vary. And because the internal random coins could vary, that could make the messages communicated also to vary. And also the messages received could be varied. That means, even if I fix my, say input to be x_i and say the final output that is learnt to be y ; even if I fix this, these parts, the remaining things will change, will take different values with different probability.

And that is why this $view_i$ will be a random variable, and it will take different values with different probabilities. So, that means, we will be now talking about probability distribution over the $view_i$, because $view_i$ will not be just taking a single value, because, even if the input and output is fixed, the messages communicated and received will be different with different probability.

(Refer Slide Time: 26:28)

Generic MPC Protocol : Privacy Definition

□ view_i: "Everything" that P_i has during the protocol --- input, output, random coins, messages sent and received



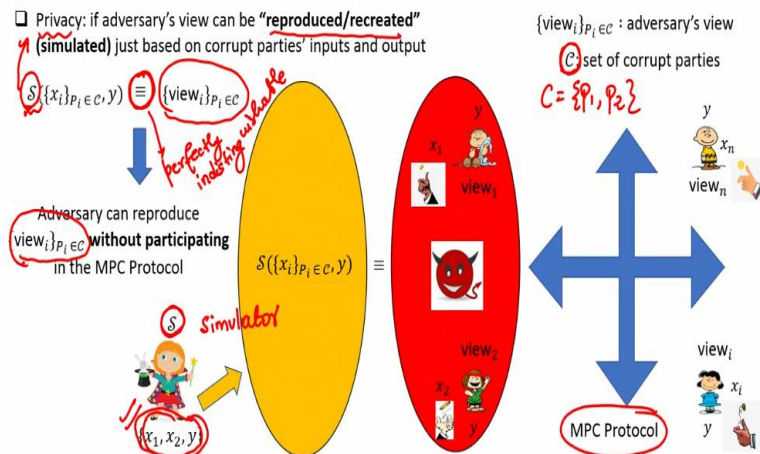
Now, imagine that, during the protocol execution, adversary goes and corrupt a set of parties in \mathcal{C} , where of course this cardinality of \mathcal{C} will be t ; it could be any set of t parties. Then what will be the view of the adversary? Well, the view of the adversary will be the collective view of all the parties, P_i , which are controlled by the adversary. So, for instance, if $t = 2$, and if adversary goes and controls say the first and the second party, then, whatever is there in $view_1$, along with $view_2$, will be called as the view of the adversary.

(Refer Slide Time: 27:13)

Generic MPC Protocol : Privacy Definition

□ view_i: "Everything" that P_i has during the protocol --- input, output, random coins, messages sent and received

□ Privacy: if adversary's view can be "reproduced/recreated" (simulated) just based on corrupt parties' inputs and output



Now, when do we say that MPC protocol satisfies the privacy property? Intuitively, what we want to capture here is the following: We want to capture that whatever adversary has seen by corrupting a set of t parties, that does not help the adversary to learn anything additional about the inputs of the other parties; of course, the adversary through the view of corrupt parties will

have the inputs of those bad guys and the function output, because that is a part of the view of the corrupt parties also.

We want to formally capture that, the view of the adversary should not reveal anything additional about the inputs of the other parties. Now, in the view of the corrupt parties, we also have the messages which those corrupt parties would have received from the honest parties, because they are also part of the view of the adversary. Intuitively, we will say that the MPC protocol is private if whatever can be learnt by the inputs and output of the corrupt parties, essentially, the same information is present in view itself, of the adversary.

That means, adversary's view can be reproduced or recreated just based on corrupt parties' input and output. And view contains the messages which honest parties would have communicated to the corrupt parties. So, what we are basically trying to argue here is that whatever honest parties would have communicated to the corrupt parties, and which is present as part of the view of the adversary, if it could be reproduced or recreated just based on the bad inputs; bad input means, the inputs of the bad guys; and the function output, then that in essence shows that whatever values the honest parties have actually sent to the bad guys during the MPC protocol execution is of no use for the bad guys.

Because, if it could be recreated just based on the bad inputs and the function output, then what is the whole point of trying to analyse the information which honest parties have communicated to the bad guys? That information can be recreated by the adversary itself, without even getting those values from the honest parties. So, how do we formalise this intuition? We formalise this intuition by saying that, for every subset \mathcal{C} of corrupt parties where the cardinality of the corrupt parties is t ; say for instance, in this case \mathcal{C} is equal to P_1 and P_2 .

So, for every subset of t bad parties, there should be some magical algorithm which we call as simulator, which can reproduce the view of the corrupt parties, just based on the inputs of the bad guys and the function output. That means, this simulator will be some algorithm which should be just given the inputs of the bad guys and the function output. And then, there will be some steps for the simulator and simulator will produce some output.

The simulator will be a randomised algorithm and its output also will be a random variable, which will have some probability distribution. So, if we can show that, if there exists a simulator which produces a probability distribution which is identical to the view of the corrupt parties in the MPC protocol, then that basically shows that my MPC protocol has the privacy property.

And why so? Because, this $view_i$ with respect to all the parties in \mathcal{C} , is actually the information which adversary possess by participating in the MPC protocol. It will have whatever decisions the bad parties have taken in the MPC protocol plus the messages which the bad parties would have received from the honest guys.

I am saying that, if all the messages which honest parties would have communicated to the bad guys, if their probability distribution could be simulated or if those messages could be regenerated just based on the inputs of the bad guys and the function output, then, that is equivalent to showing that whatever communication has happened from the honest parties to the bad guy is of no use to the bad guys.

And that shows that my protocol ensures the privacy property. And indeed, if I show the existence of such a simulator for which these 2 probability distributions are equivalent, then in essence, I am basically showing that adversary can regenerate its view of the MPC protocol by itself performing the role of the simulator, and without participating in the MPC protocol.

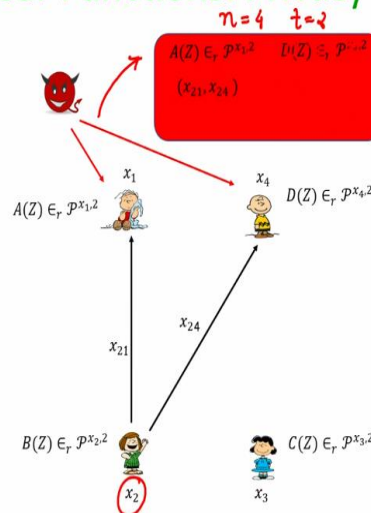
That means, from the viewpoint of the MPC protocol, participating in the MPC protocol is as good as not participating in the MPC protocol and just stick to the corrupt parties' input, function output and run the simulator and come up with the messages which honest parties would have communicated to the bad guys. So, that precisely is the essence of this definition.

So, to formally put, we will say that a generic MPC protocol has the privacy property if for every subset \mathcal{C} of bad parties corrupting up to t parties, the view of the adversary with respect to the parties in \mathcal{C} , can be simulated by a simulator such that the output of the simulator where the simulator is just given the inputs of the parties in \mathcal{C} and the function output; and the output of the simulator, its probability distribution should be perfectly indistinguishable from the view of the bad guys, in the real MPC protocol.

So, this notation here means perfectly indistinguishable. If this is the case, then we will say that our MPC protocol does not reveal any additional information about the inputs of the honest parties. That means, whatever messages honest parties have communicated to the bad guys, it does not reveal anything additional, because they can be simulated by the simulator itself.

(Refer Slide Time: 34:40)

BGW Protocol for Linear Functions: Privacy

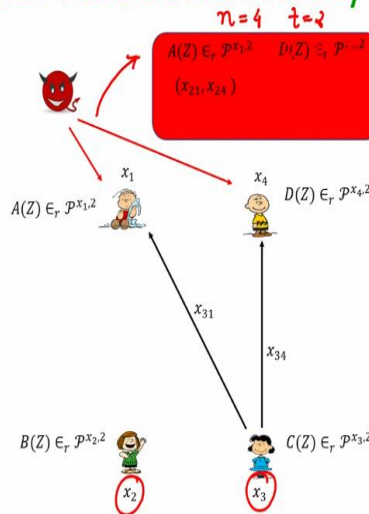


So, let us now try to prove that the BGW MPC protocol for the linear function satisfies this privacy property. And again, for simplicity, I am taking the case where $n = 4, t = 2$ and my set of corrupt parties in \mathcal{C} are P_1 and P_4 . So, let us see what will be the view of the adversary, namely, the collective view of party 1 and party 4. So, the sharing polynomials which are picked by the party 1 and party 4, they will be a part of the view of the adversary.

So, $t = 2$, that means, all the sharing polynomials are of degree-2. So, the A polynomial and the D polynomial will be part of the view of the adversary. Now, the shares P_2 and P_3 communicated to P_1 and P_4 during the protocol execution will also be a part of the view. So, regarding the second party's input, P_1 gets the share x_{21} ; so, that will be included in adversary's view. And x_{24} will be communicated to party P_4 ; so, that also will be a part of the view.

(Refer Slide Time: 36:02)

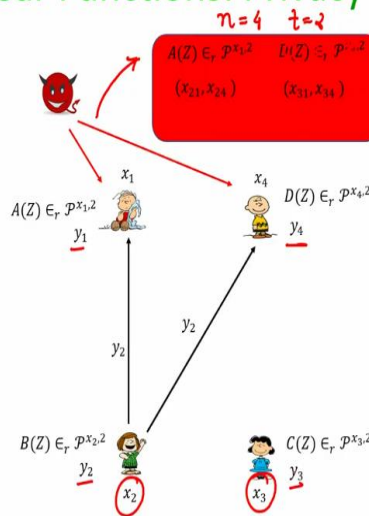
BGW Protocol for Linear Functions: Privacy



And for the third party's input, the adversary receives the first share and the fourth share. So, that will be a part of the view. What else?

(Refer Slide Time: 36:11)

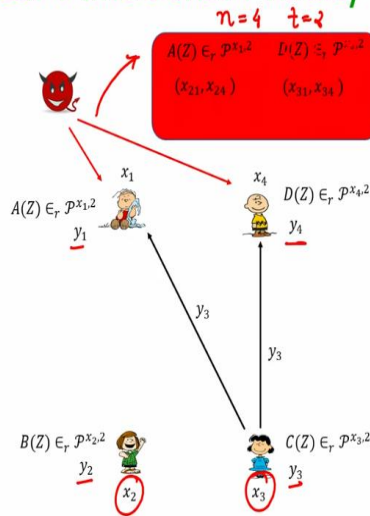
BGW Protocol for Linear Functions: Privacy



So, now, everyone computes locally their respective shares of y . And then, P_2 would have announced its share of y .

(Refer Slide Time: 36:22)

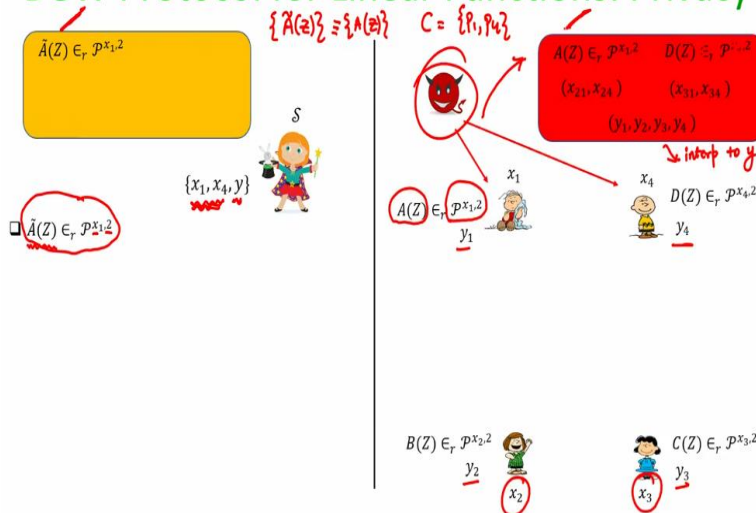
BGW Protocol for Linear Functions: Privacy



And P_3 would have announced its shares of y .

(Refer Slide Time: 36:28)

BGW Protocol for Linear Functions: Privacy



All together, these shares of y , namely, the shares of y with respect to 1, 2, 3 and 4 will now be public. And hence, it will be added to the view of adversary. And adversary will know that, okay, this vector of y shares will interpolate to y . That is the view of this adversary by participating in the MPC protocol. Now, we have to show that, through simulator, we have to argue here that, whatever information the adversary would have obtained by participating in the MPC protocol and by corrupting P_1 and P_4 , could be regenerated by a simulator who is only given the inputs P_1, P_4 and the function output $+y$.

Why P_1, P_4 ? Because we have to reproduce the view with respect to the set of parties in \mathcal{C} ; and we are fixing the set of parties in \mathcal{C} to be P_1 and P_4 . So, we will say to the simulator, okay,

assume simulator, you have, you are given the values x_1, x_4 , the inputs of the corrupt parties and the final output y ; can you reproduce whatever information this adversary would have actually seen by participating in the MPC protocol?

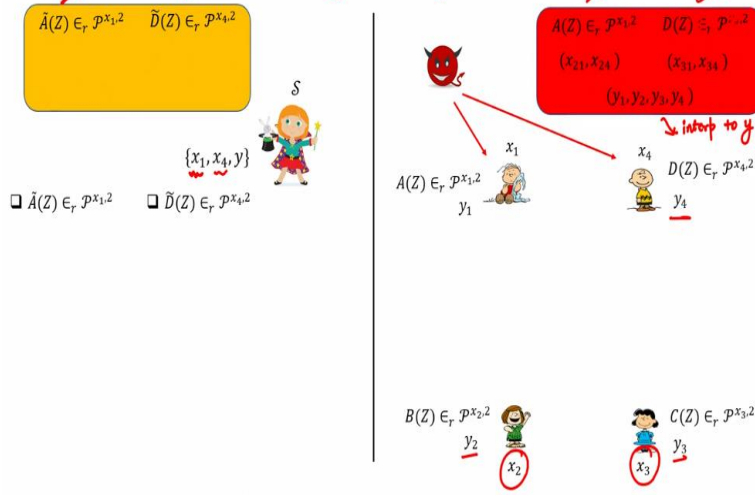
So, the first thing this simulator can easily do is the following: It has to regenerate the A polynomials and the D polynomials which P_1 and P_4 would have used for secret-sharing their x_1 and x_4 . And that is very easy to reproduce. What simulator can do is the following: It can just randomly pick a t -degree polynomial; t in this case is 2. I am calling that randomly picked polynomial as \tilde{A} , because that could be a different polynomial from the exact A polynomial which P_1 used in the real execution.

And the only constraint on this polynomial is that its constant term should be 1. Now, my claim is that, if you take the probability distribution of this simulated \tilde{A} polynomial, then that has the same probability distribution as the A polynomial which was actually picked by P_1 . Why so? Because, this A polynomial would have been a random polynomial from this set of all possible 2-degree polynomials whose constant term is x_1 , and so is the case for this \tilde{A} polynomial as well.

It is also a randomly chosen 2-degree polynomial whose constant term is x_1 . And that is why, I can say that, component wise, this component here, and this component here, have the identical probability distributions. That means, with whatever probability $\tilde{A}(Z)$ polynomial can take values, with same probability, my $A(Z)$ polynomial can also take values. And this basically captures the following: The simulator, it knows its own input.

(Refer Slide Time: 39:52)

BGW Protocol for Linear Functions: Privacy

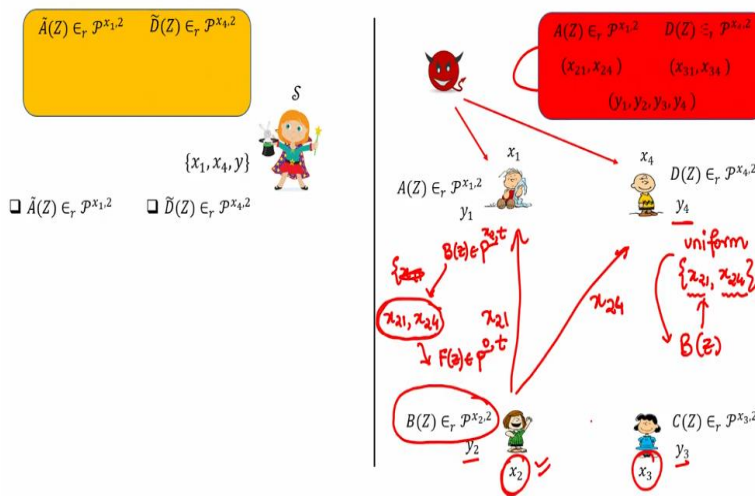


Its own input means, imagine the case when simulator is the adversary itself; so, it knows its own input x_1 and input x_4 , and it knows the way by which it has secret-shared x_1 and x_4 . So, that is what I am basically doing here. It knows that I have secret-shared the inputs x_1 by picking a random 2-degree polynomial whose constant is x_1 , and I have randomly shared x_4 by picking a random D polynomial with degree-2 whose constant term is x_4 .

So, these 2 components can be easily simulated, regenerated. That does not require the help of interaction with honest parties and so on. So, the first 2 components are regenerated with identical probability distribution; fine; but now comes the difficult part.

(Refer Slide Time: 40:46)

BGW Protocol for Linear Functions: Privacy



In the real execution, regarding the second party's input, adversary would have obtained the shares x_{21} and x_{24} . By the way, I stress that there is a probability distribution associated with

x_{21} and x_{24} , because, even if x_2 is fixed once for all, the values x_{21} and x_{24} can take different values with different probability, because this $B(Z)$ polynomial would have picked uniformly at random.

It will not be the case that the same B polynomial is picked again and again and again by P_2 for sharing its same input x_2 . And that is why, depending upon what is the value of $B(Z)$ polynomial, the values of x_{21} and x_{24} are determined. So, that means, there is a probability distribution associated with x_{21} and x_{24} . But what is that probability distribution?

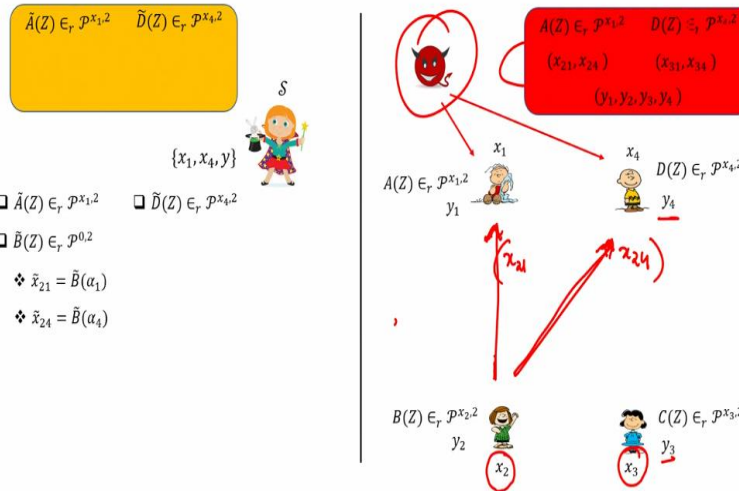
My claim here is that, the probability distribution on this x_{21} and x_{24} is uniform probability distribution. That means, these x_{21} and x_{24} could be any random field elements, they are not dependent on B polynomial. It could be any B polynomial; it could be in fact, any t -degree polynomial whose constant term could be anything; when evaluated at α_1 , produces x_{21} , and when evaluated at α_4 , would have been x_{24} ; because this comes from your privacy property of Shamir secret-sharing.

When we discussed the privacy property of Shamir secret-sharing, there we argued that, you take any t values from the field, they could lie on any random t -degree polynomial whose constant term could be any value from the field. That means, it could be the case that this x_{21} and x_{24} which adversary has seen, it is coming because of a polynomial $B(Z)$ belonging to, say, set of all polynomials whose constant term is x_2 and degree is t .

Or it could be equally the case that these 2 values are shares for a polynomial say $F(Z)$, belonging to the set of all polynomials whose constant term is 0, say, and degree is t . That is also possible. That means, we cannot say that these 2 shares x_{21} and x_{24} cannot occur as 2 shares for secret being x_2 or it cannot occur as shares for the secret being 0 and so on, because they are just 2 random shares on a 2-degree random polynomial whose constant term is an unknown value.

(Refer Slide Time: 44:04)

BGW Protocol for Linear Functions: Privacy

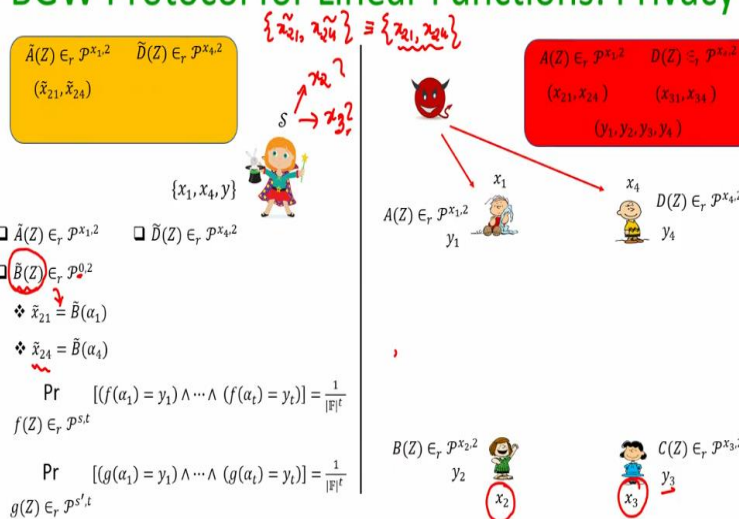


So, that means, if I consider from adversary's viewpoint, he has seen the value x_{21} and x_{24} . From its viewpoint, this x_{21} and x_{24} could be the shares for secret x_2 , or it could be the shares for value 0, or it could be the shares for any other field element and so on. That means, it knows beforehand itself that, okay, the 2 shares that it is going to receive could be the shares for any t -degree random polynomial whose constant term could be any value from the field.

That means, it does not help him to learn anything concretely about x_2 . x_2 could be as good as 0 as well, from the adversary's viewpoint. And that is what we use here to simulate these 2 shares by the simulator. So, what the simulator has to do? The goal of the simulator, has to reproduce a probability distribution which is identical to this x_{21} and x_{24} . How it does that?

(Refer Slide Time: 45:08)

BGW Protocol for Linear Functions: Privacy

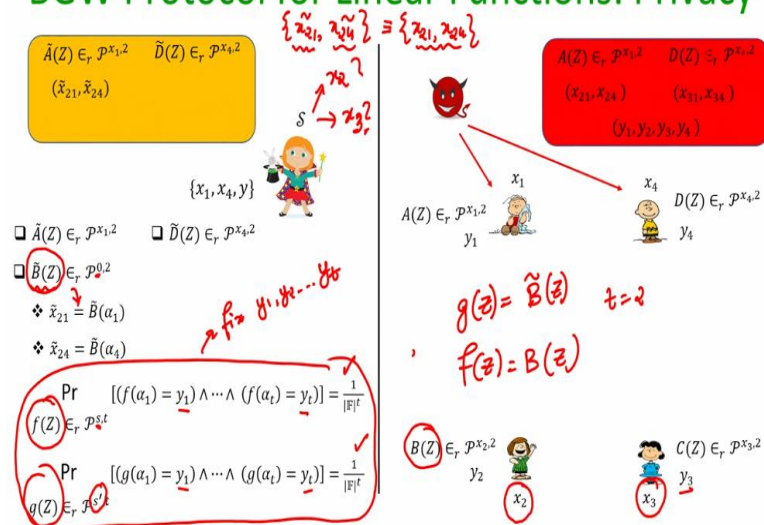


It randomly picks a B polynomial whose constant term is 0. Why 0? Because it does not know the value of x_2 , the concrete value of x_2 . It does not know the concrete value of x_2 and x_3 . It is just pretending as if x_2 is 0, and it is playing the role of an honest party P_2 by itself acting as a P_2 and acting as a dealer. That means, it is acting as if P_2 wants to secret-share the value 0.

And for that it is randomly picking a polynomial \tilde{B} whose constant term is 0, computing the shares x_{21} and x_{24} as per this sharing polynomial, and writing down it in the simulated view. And my claim is that, if I take the probability distribution of this simulated \tilde{x}_{21} and \tilde{x}_{24} its probability distribution is identical to the actual x_{21} and x_{24} , which the adversary would have seen in the real protocol; because this x_{21} and x_{24} which adversary would have seen in the MPC protocol are just 2 random shares on a random 2-degree polynomial whose constant term is unknown, and they could take the value \tilde{x}_{21} and \tilde{x}_{24} also with equal probability; that is precisely is the intuition here.

(Refer Slide Time: 46:37)

BGW Protocol for Linear Functions: Privacy



So, this is precisely the reason because of which this simulation strategy works. Recall: When we discussed the properties of t -degree polynomials over field, there we argued that it does not matter whether you are picking a random polynomial for sharing the value s or you are picking a random polynomial for sharing the value s' ; with equal probability, any t shares could take the values y_1, \dots, y_t .

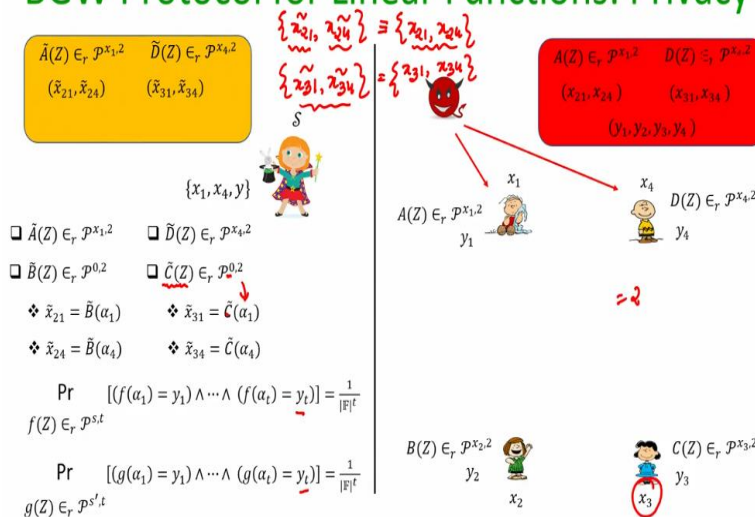
That means, if you fix y_1, y_2, \dots, y_t , they could be shares for sharing the value s as well as for sharing the value s' with equal probability, provided the underlying sharing polynomials are

picked uniformly at random. And that is what the simulator is doing. So, $f(Z)$ was actually your $B(Z)$ polynomial. Set $f(Z) = B(Z)$, and set $g(Z) = \tilde{B}(Z)$, and $t = 2$. And I am focusing on x_{21} and x_{24} .

These x_{21} and x_{24} could occur as the shares of 0 as well as the share of x_2 . And in the same way, this \tilde{x}_{21} and \tilde{x}_{24} , they could also be the shares of 0 or they could also be the shares of x_2 with equal probability. And hence, the probability distribution of the actual x_{21} and x_{24} which adversary would have seen in the real view in the MPC protocol, is exactly identical to the simulated \tilde{x}_{21} and \tilde{x}_{24} which the simulator has generated.

(Refer Slide Time: 48:26)

BGW Protocol for Linear Functions: Privacy

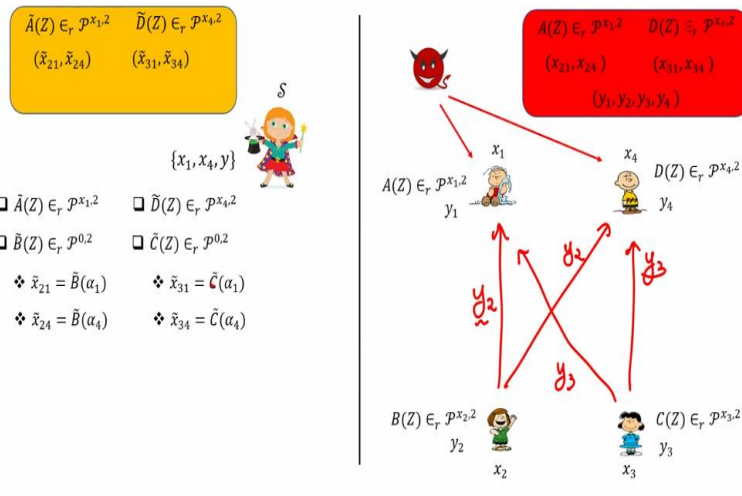


In the same way, in the same argument, what simulator is doing is, it is just randomly picking a C polynomial whose constant term is 0, and as per that sharing polynomial, producing the simulated x_{31} and x_{34} . And whatever argument we used just now, we can use the same argument to argue that the simulated x_{31} and x_{34} , their probability distribution is same as the actual x_{31} and x_{34} that adversary has seen in the real view.

This is because x_{31} and x_{34} could occur as the share of x_3 as well as $x_3 = 0$, with equal probability. And \tilde{x}_{31} and \tilde{x}_{34} could also occur as the share for 0 as well as for the actual x_3 , with equal probability. And hence, till now, whatever the simulator has simulated, it follows that its probability distribution is identical to the corresponding component in the real view. Till now, everything is fine. Now, what about the vector of y output shares? How it can be simulated?

(Refer Slide Time: 49:52)

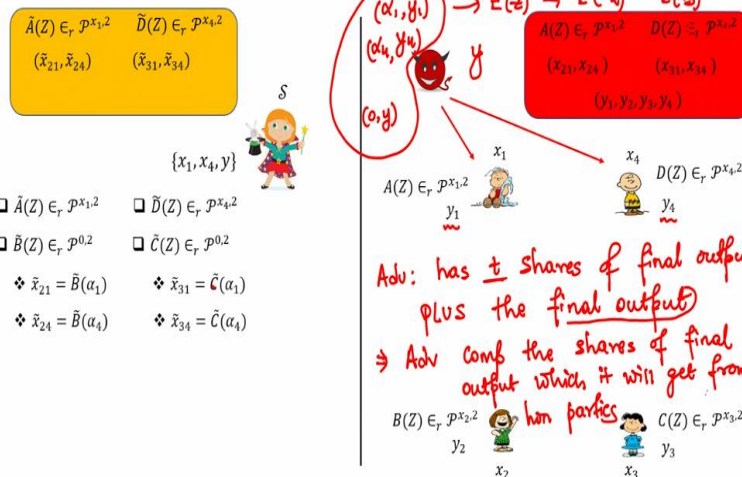
BGW Protocol for Linear Functions: Privacy



That means, in the real view; real view means, in the real MPC protocol; adversary would have seen that P_2 has communicated y_2 , and it would have seen that P_3 would have communicated y_3 . And it has its own y_1, y_4 . Now, simulator has to simulate y_2 and y_3 . Now, here comes the crux of the simulator. See, in the real MPC protocol, will learning y_2 and y_3 from P_2 and P_3 reveal anything additional to the adversary? The answer is no.

(Refer Slide Time: 50:36)

BGW Protocol for Linear Functions: Privacy

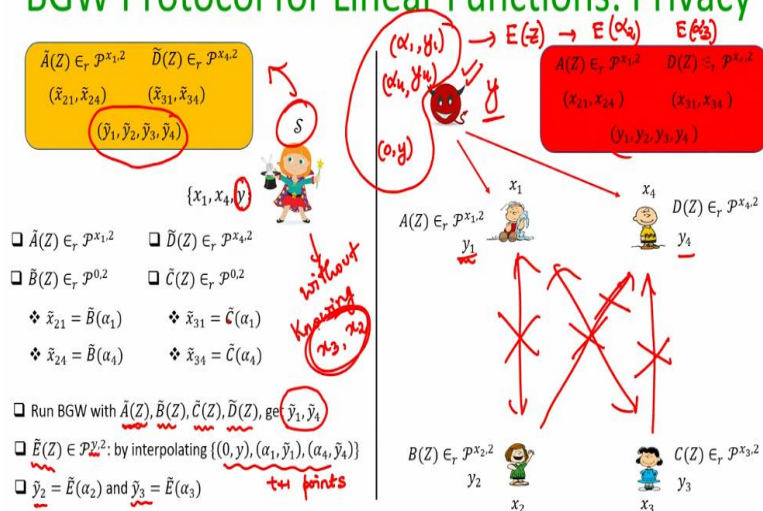


Why so? Because, adversary has already 1 share of y and another share of y , and anyhow it will learn the function output y . So, adversary has t shares of final output plus the final output itself. My claim is that through this, adversary can itself compute the shares of final output which it will get from honest parties. Why so? Because it already has t shares, and plus the

final output; and remember the final output is nothing but the value $(0, y)$; that constitutes a point on the final polynomial which has to be interpolated, to reconstruct the function output. And anyhow it has (α_1, y_1) and (α_4, y_4) . And it knows that the final y polynomial which will be interpolated will have degree $t + 1$, in this case, 3. So, it has already 3 points on the final polynomial which will be interpolated. So, it can itself interpolate that E polynomial. And if it itself can interpolate the polynomial, it will itself know that what is the value of $E(\alpha_2)$ which is y_2 and $E(\alpha_3)$ which is y_3 , which P_2 and P_3 would have communicated to the adversary. That means, sending y_2 from P_2 and sending y_3 from P_3 to the adversary is not a new information which adversary will have.

(Refer Slide Time: 53:04)

BGW Protocol for Linear Functions: Privacy



It will already know, based on its shares y_1, y_4 and the function output y which it will anyhow learn, that, okay, these are the y_3 and y_2 which I will be getting. And that is the precise intuition which we can use when we simulate the output vectors y_1, y_2, y_3, y_4 by the simulator. So, what the simulator will do is the following: It will run the BGW protocol in its mind, assuming that P_1 has used this sharing polynomial, P_2 has used this sharing polynomial, P_3 has used this sharing polynomial and P_4 has used this sharing polynomial.

And it will get the corresponding first share and fourth share. I am denoting it as \tilde{y}_1 and \tilde{y}_4 . It can do that because it is basically just playing the role of P_1 and P_4 in its mind. And now what it does is the following: Remember, the simulator also has the final function output. So, it has now together $t + 1$ points through which it can interpolate a t -degree polynomial whose constant term will be y .

And now, it can say the following: That okay, this is the share of y which P_2 will be communicating, and this is the share of y which P_3 will be communicating. And that will be the simulated vector of shares of y ; $\widetilde{y}_1, \widetilde{y}_2, \widetilde{y}_3, \widetilde{y}_4$. And now, it is easy to see that this $\widetilde{y}_1, \widetilde{y}_2, \widetilde{y}_3, \widetilde{y}_4$, it has the same distribution as the actual y_1, y_2, y_3, y_4 , which adversary sees in the real execution.

For both the vectors, the constant term of the interpolated polynomial will be y . For both the vectors, the first component and the fourth component are actually what adversary would have seen if they would have participated in the MPC protocol. And now, you can see that this simulator, it is able to recreate whatever information the adversary will have by participating in the MPC protocol, without knowing; that is important; without knowing x_2 and x_3 .

That means, even without knowing x_2 and x_3 , it could reproduce whatever information this adversary would have obtained by interacting with x_2 and x_3 . And then, in essence, this is equivalent to showing that this interaction with x_2 and this interaction with x_3 is completely useless for the adversary. Similarly, this interaction between 2 and 4 and interaction between 3 and 4 is completely useless for the adversary; because, whatever could be obtained by the adversary by interaction, it could be reproduced without even knowing x_2 and x_3 .

(Refer Slide Time: 56:15)

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

And that shows that the BGW MPC protocol for the linear function satisfies the formal privacy definition that we have practiced. Thank you.