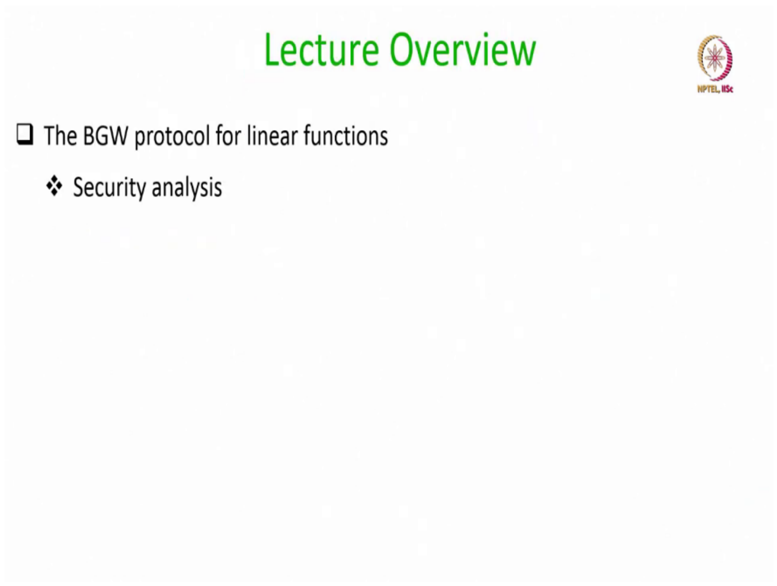
Secure Computation: Part II

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

Lecture - 45
The BGW MPC Protocol for Linear Functions: Security Analysis

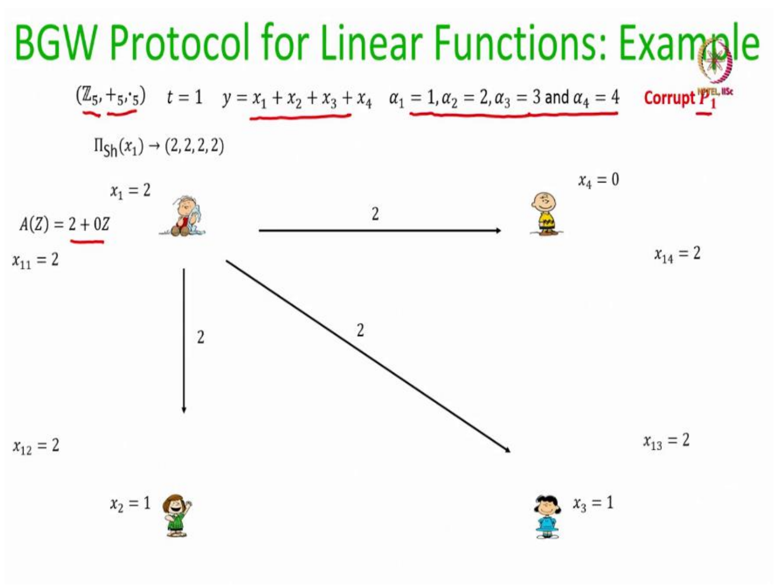
(Refer Slide Time: 00:30)



Lecture Overview

- The BGW protocol for linear functions
  - Security analysis

(Refer Slide Time: 00:32)



BGW Protocol for Linear Functions: Example

$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ Corrupt P_1

$\Pi_{Sh}(x_1) \rightarrow (2, 2, 2, 2)$

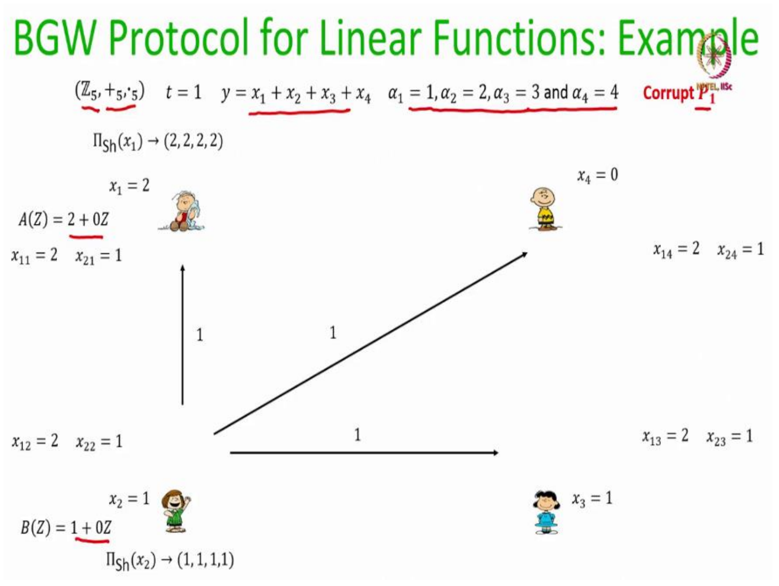
Hello everyone, welcome to this lecture. So, in this lecture, we will quickly do a security analysis for the BGW MPC Protocol for Linear Functions which we had discussed in the previous lecture. So, before going to the formal analysis, we try to understand why this protocol is secure with an example here.

So, let us consider a very simple function with the addition of four inputs here. We have four parties and up to one party can be corrupt. So, for simplicity, imagine that the first party is corrupt; of course, the parties will not be knowing that it is the first party who is corrupt, but they will be knowing that up to one party can be corrupt here.

And we will perform all the computations over the field $\mathbb{Z}_5$, where the plus operation is the addition modulo 5 operation, and the multiplication operation is the multiplication modulo 5 operation. We fix the evaluation points which are going to be used during the instances of the VSS to be 1, 2, 3 and 4 for P_1, P_2, P_3, P_4, respectively.

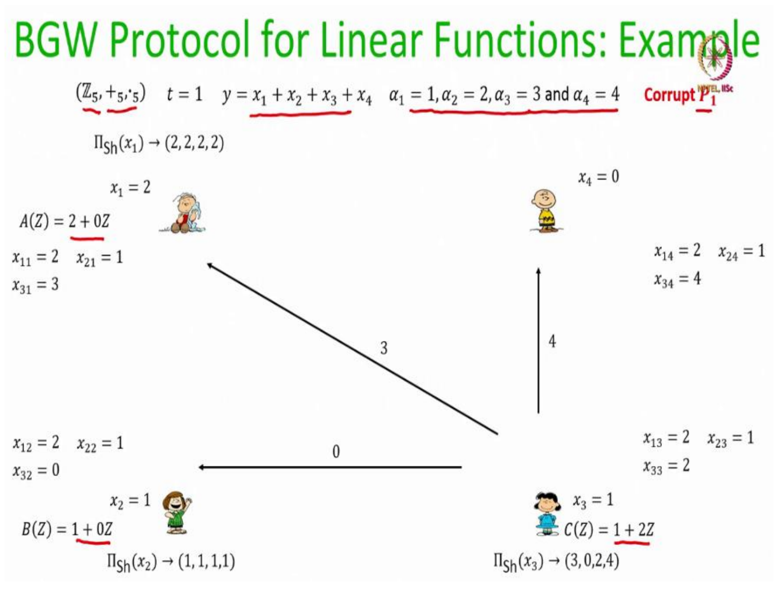
Now, suppose the inputs of the parties are 2, 1, 1 and 0 respectively. So, during the input stage, suppose the Shamir sharing polynomial through which x_1 is shared is this polynomial A, which is $2 + 0Z$. And say at the end of the sharing phase protocol of the VSS scheme the shares for the parties are 2, 2, 2 and 2 respectively.

(Refer Slide Time: 02:03)


BGW Protocol for Linear Functions: Example
$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ Corrupt P_1
$\Pi_{Sh}(x_1) \rightarrow (2,2,2,2)$
$x_1 = 2$
$A(Z) = 2 + 0Z$
$x_{11} = 2$ $x_{21} = 1$
$x_4 = 0$
$x_{14} = 2$ $x_{24} = 1$
1
1
1
$x_{12} = 2$ $x_{22} = 1$
$x_{13} = 2$ $x_{23} = 1$
$x_2 = 1$
$B(Z) = 1 + 0Z$
$\Pi_{Sh}(x_2) \rightarrow (1,1,1,1)$
$x_3 = 1$

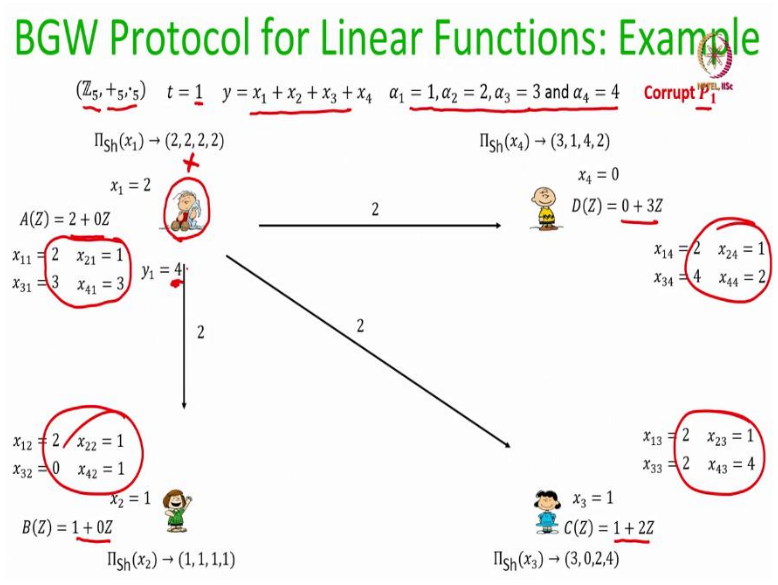
In parallel, suppose P_2 in its instance of the secret sharing protocol where VSS protocol uses the Shamir sharing polynomial $1 + 0Z$ and say the resultant shares are 1, 1, 1, 1 respectively at the end of the sharing phase.

(Refer Slide Time: 02:20)


BGW Protocol for Linear Functions: Example
$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ Corrupt P_1
$\Pi_{Sh}(x_1) \rightarrow (2,2,2,2)$
$x_1 = 2$
$A(Z) = 2 + 0Z$
$x_{11} = 2$ $x_{21} = 1$
$x_{31} = 3$
$x_4 = 0$
$x_{14} = 2$ $x_{24} = 1$
$x_{34} = 4$
3
4
0
$x_{12} = 2$ $x_{22} = 1$
$x_{32} = 0$
$x_{13} = 2$ $x_{23} = 1$
$x_{33} = 2$
$x_2 = 1$
$B(Z) = 1 + 0Z$
$\Pi_{Sh}(x_2) \rightarrow (1,1,1,1)$
$x_3 = 1$
$C(Z) = 1 + 2Z$
$\Pi_{Sh}(x_3) \rightarrow (3,0,2,4)$

In the same way imagine that P_3 during its instance of the VSS scheme has used this Shamir sharing polynomial resulting in the shares 3, 0, 2 and 4.

(Refer Slide Time: 02:37)

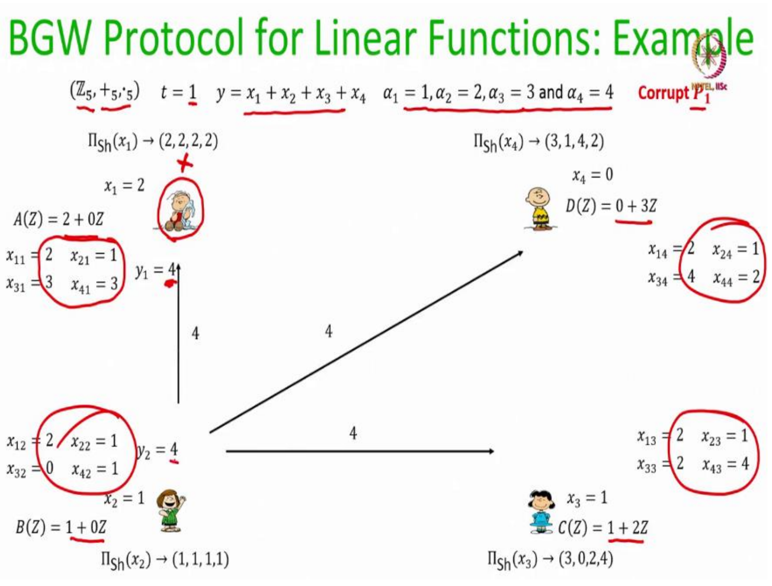

BGW Protocol for Linear Functions: Example
$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ Corrupt P_1
$\Pi_{Sh}(x_1) \rightarrow (2,2,2,2)$
$\Pi_{Sh}(x_4) \rightarrow (3,1,4,2)$
$x_1 = 2$
$A(Z) = 2 + 0Z$
$x_{11} = 2$ $x_{21} = 1$
$x_{31} = 3$ $x_{41} = 3$
$y_1 = 4$
$x_4 = 0$
$D(Z) = 0 + 3Z$
$x_{14} = 2$ $x_{24} = 1$
$x_{34} = 4$ $x_{44} = 2$
2
2
2
$x_{12} = 2$ $x_{22} = 1$
$x_{32} = 0$ $x_{42} = 1$
$x_{13} = 2$ $x_{23} = 1$
$x_{33} = 2$ $x_{43} = 4$
$x_2 = 1$
$B(Z) = 1 + 0Z$
$\Pi_{Sh}(x_2) \rightarrow (1,1,1,1)$
$x_3 = 1$
$C(Z) = 1 + 2Z$
$\Pi_{Sh}(x_3) \rightarrow (3,0,2,4)$

And P_4 in its instance of the VSS scheme uses the polynomial $0 + 3Z$ resulting in the vector of shares 3, 1, 4, 2. So, all this VSS instances uses they use polynomials of degree 1 because t is equal to 1. Now, since there is only one linear gate namely the addition of all the four inputs, each party will locally add the shares of x_1, x_2, x_3, x_4 to obtain the share of y.

So, P_1 is going to add all these four shares; and of course, the addition is performed modulo 5 that will result in P_1's share to be 4. And in the same way $P_2's$ share will be the addition of all these four values, P_3's share will be the addition of these four shares and P_4's share will be the addition of these four shares. And now the output starts. So, since P_1 is corrupt, it is supposed to make the share 4 public.

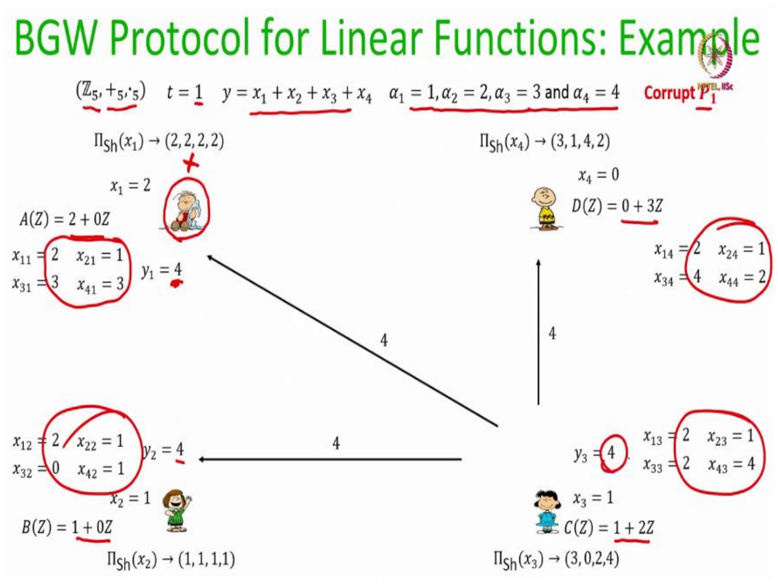
But suppose it says that my share is 2 and that is fine because that is allowed because P_1 is corrupt. So, it can deviate from the protocol instructions. So, it is supposed to make public the correct share of y, but it is making an incorrect share of y public. No one of course, will know whether P_1 is corrupt or not.

(Refer Slide Time: 04:06)



BGW Protocol for Linear Functions: Example
$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ Corrupt P_1
$\Pi_{Sh}(x_1) \rightarrow (2,2,2,2)$
$x_1 = 2$
$A(Z) = 2 + 0Z$
$x_{11} = 2$ $x_{21} = 1$
$x_{31} = 3$ $x_{41} = 3$
$y_1 = 4$
$\Pi_{Sh}(x_4) \rightarrow (3,1,4,2)$
$x_4 = 0$
$D(Z) = 0 + 3Z$
$x_{14} = 2$ $x_{24} = 1$
$x_{34} = 4$ $x_{44} = 2$
4
4
4
$x_{12} = 2$ $x_{22} = 1$
$x_{32} = 0$ $x_{42} = 1$
$y_2 = 4$
$x_2 = 1$
$B(Z) = 1 + 0Z$
$\Pi_{Sh}(x_2) \rightarrow (1,1,1,1)$
$x_{13} = 2$ $x_{23} = 1$
$x_{33} = 2$ $x_{43} = 4$
$x_3 = 1$
$C(Z) = 1 + 2Z$
$\Pi_{Sh}(x_3) \rightarrow (3,0,2,4)$

(Refer Slide Time: 04:12)



BGW Protocol for Linear Functions: Example

$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ Corrupt P_1

$\Pi_{Sh}(x_1) \rightarrow (2,2,2,2)$ $x_1 = 2$ $A(Z) = 2 + 0Z$ $x_{11} = 2$ $x_{21} = 1$ $x_{31} = 3$ $x_{41} = 3$ $y_1 = 4$

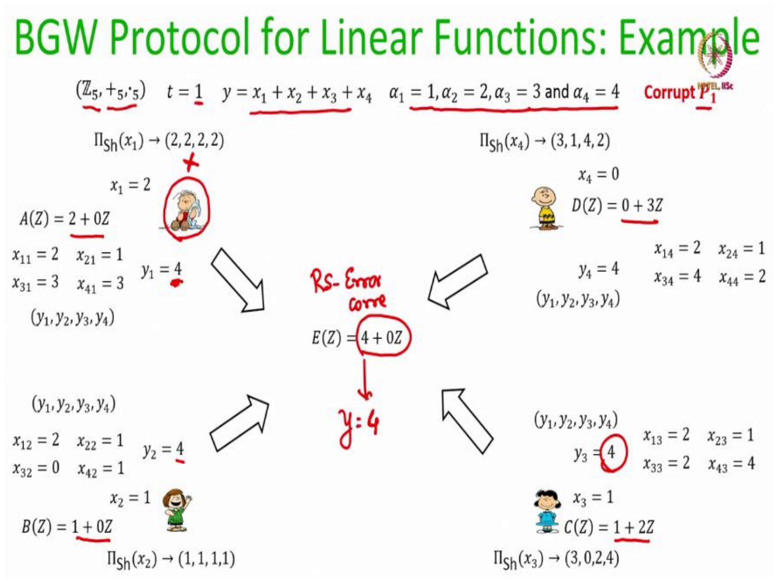
$\Pi_{Sh}(x_4) \rightarrow (3,1,4,2)$ $x_4 = 0$ $D(Z) = 0 + 3Z$ $x_{14} = 2$ $x_{24} = 1$ $x_{34} = 4$ $x_{44} = 2$

$x_{12} = 2$ $x_{22} = 1$ $x_{32} = 0$ $x_{42} = 1$ $y_2 = 4$ $x_2 = 1$ $B(Z) = 1 + 0Z$ $\Pi_{Sh}(x_2) \rightarrow (1,1,1,1)$

$y_3 = 4$ $x_{13} = 2$ $x_{23} = 1$ $x_{33} = 2$ $x_{43} = 4$ $x_3 = 1$ $C(Z) = 1 + 2Z$ $\Pi_{Sh}(x_3) \rightarrow (3,0,2,4)$

However, the other parties correctly make their shares public. So, P_3 makes public the share 4, P_4 makes the share public.

(Refer Slide Time: 04:21)



BGW Protocol for Linear Functions: Example

$(\mathbb{Z}_5, +_5, \cdot_5)$ $t = 1$ $y = x_1 + x_2 + x_3 + x_4$ $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3$ and $\alpha_4 = 4$ Corrupt P_1

$\Pi_{Sh}(x_1) \rightarrow (2,2,2,2)$ $x_1 = 2$ $A(Z) = 2 + 0Z$ $x_{11} = 2$ $x_{21} = 1$ $x_{31} = 3$ $x_{41} = 3$ $y_1 = 4$ (y_1, y_2, y_3, y_4)

$\Pi_{Sh}(x_4) \rightarrow (3,1,4,2)$ $x_4 = 0$ $D(Z) = 0 + 3Z$ $x_{14} = 2$ $x_{24} = 1$ $x_{34} = 4$ $x_{44} = 2$ $y_4 = 4$ (y_1, y_2, y_3, y_4)

RS- Error corre

$E(Z) = 4 + 0Z$

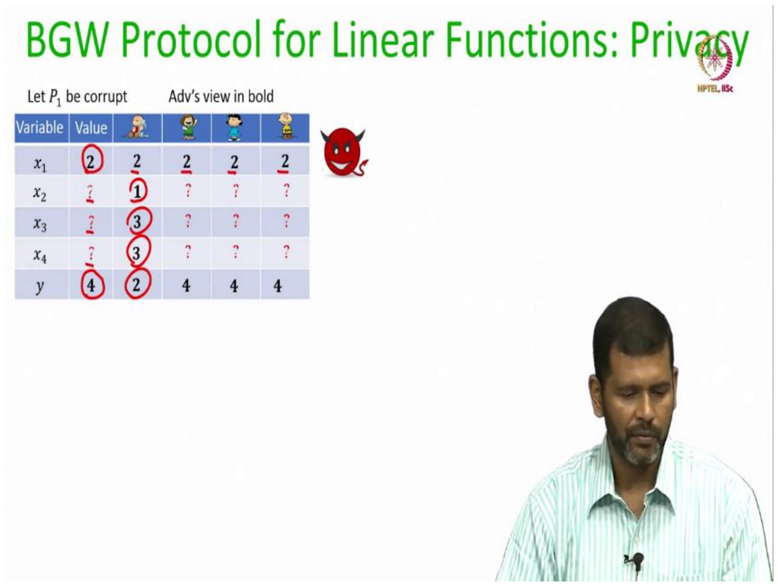
$y:4$

(y_1, y_2, y_3, y_4) $x_{12} = 2$ $x_{22} = 1$ $x_{32} = 0$ $x_{42} = 1$ $y_2 = 4$ $x_2 = 1$ $B(Z) = 1 + 0Z$ $\Pi_{Sh}(x_2) \rightarrow (1,1,1,1)$

(y_1, y_2, y_3, y_4) $y_3 = 4$ $x_{13} = 2$ $x_{23} = 1$ $x_{33} = 2$ $x_{43} = 4$ $x_3 = 1$ $C(Z) = 1 + 2Z$ $\Pi_{Sh}(x_3) \rightarrow (3,0,2,4)$

And now, the vector of shares corresponding to the output gate will be available with everyone. And now they will do the Reed Solomon error correction and recover this 1-degree polynomial. And the output will be $y = 4$, that is a possible execution for the BGW protocol.

(Refer Slide Time: 04:44)



BGW Protocol for Linear Functions: Privacy

Let P_1 be corrupt   Adv's view in bold

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

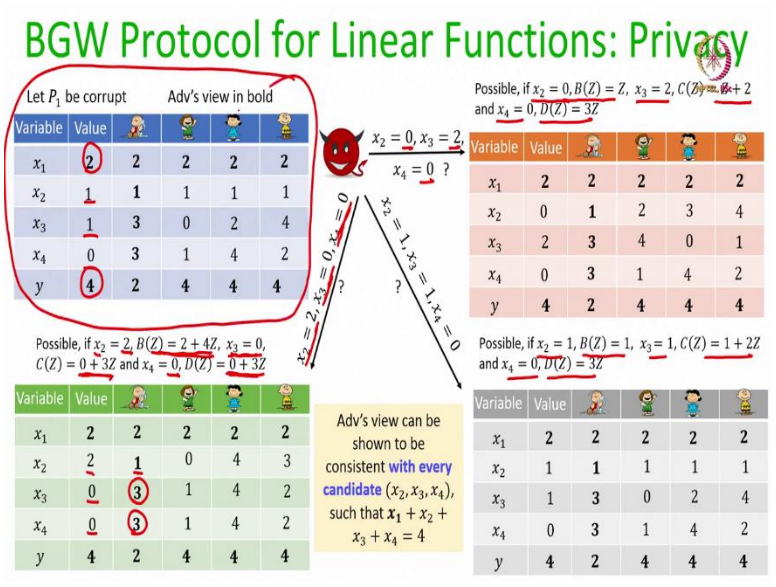
Now, let us see that whether the privacy property holds here or not; that means, whether the adversary learns anything additional about the inputs of the honest parties, beyond what it is allowed to learn. So, in this table I have highlighted the view of the adversary in bold, what does the view mean? View means whatever information the adversary learns through the corrupt party.

So, in this case P_1 is the only corrupt party. So, the adversary's view will be consisting of the input of the party P_1. And all the shares corresponding to this input are generated during the VSS instance where P_1 is the dealer. However, for the remaining three VSS instances, the view of the adversary will be independent of what exactly are the secrets in those VSS instances.

So, that is why x_2 is unknown for the adversary, x_3 is unknown for the adversary, x_4 is unknown for the adversary. And the adversary will have only its share from those VSS instances. And of course, it will not know what shares the honest parties have received in those VSS instances from the corresponding dealers.

And, now during the reconstruction stage everyone has made public their respective shares of the output value. Of course, P_1 has made public the incorrect share and then the Reed Solomon error correction has resulted in the output $y = 4$ that is the view of the adversary. Now, this view adversary can try to analyse to see whether it can learn anything additional about x_2, x_3, x_4.

(Refer Slide Time: 06:28)



BGW Protocol for Linear Functions: Privacy

Let P_1 be corrupt   Adv's view in bold

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

$x_2 = 0, x_3 = 2, x_4 = 0$?

$x_2 = 2, x_3 = 0, x_4 = 0$?

$x_2 = 1, x_3 = 1, x_4 = 0$?

Possible, if $x_2 = 0, B(Z) = Z$, $x_3 = 2, C(Z) =$ [illegible] $+ 2$ and $x_4 = 0, D(Z) = 3Z$

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

Possible, if $x_2 = 2, B(Z) = 2 + 4Z$, $x_3 = 0$, $C(Z) = 0 + 3Z$ and $x_4 = 0, D(Z) = 0 + 3Z$

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

Adv's view can be shown to be consistent **with every candidate** (x_2, x_3, x_4), such that $\boldsymbol{x_1} + x_2 + x_3 + x_4 = 4$

Possible, if $x_2 = 1, B(Z) = 1$, $x_3 = 1, C(Z) = 1 + 2Z$ and $x_4 = 0, D(Z) = 3Z$

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

I have filled this table with the values which have been used by the parties during the execution of the protocol in our current example. Now, this x_2, x_3, x_4 they are currently unknown for the adversary. So, adversary, based on its own input 2 and the function output $y = 4$, can make many hypotheses.

It knows that the function which is getting computed is a sum of x_1, x_2, x_3 and x_4. Out of that x_1 is known and the sum y is also known. So, it can always conclude that there are various possibilities for x_2, x_3, x_4. So, for instance one possibility could be that P_2 has executed the protocol with input 2, and P_3 has executed the protocol with input 0 and P_4 has executed the protocol with input 0, that is quite a possibility, because if indeed x_2 is 2 and x_3 is 0 and x_4 is 0 and if P_1's input would have been 2, then together it will result in the function output to be $y = 4$.

Now, what adversary can try to do is the following. This table right now is unknown for the adversary namely the values in red colour. And adversary makes a hypothesis and is it possible that I have participated in the protocol execution where x_2 was 2, x_3 was 0, x_4 was 0.

And the values in bold in this table are the values which I have seen. And indeed, it is quite possible that P_2 has participated in the protocol with x_2 equal to 2 and P_3 has participated with input x_3 equal to 0 and P_4 has participated with input x_4 equal to 0.

More specifically, if P_2 would have used the polynomial $2 + 4Z$ in an instance of the BGW protocol for linear function, then it would have resulted in P_1 getting the share 1. Of course, P_2, P_3, P_4 might get other shares, but P_1 is not going to see what exactly are the shares which 2, 3 and 4 have received in that instance.

And in the same way it could be possible that $P_3{}'$s input during the run of the BGW protocol was 0, and it has used the sharing polynomial $0 + 3Z$ which would have resulted in the share 3 for P_1. And similarly, it could be possible that $P_4{}'$s input was 0, its sharing polynomial was $0 + 3Z$ which would have resulted in P_1 getting the share 3 during a run of BGW protocol.

And now you can see that magically it's so happening here that if the BGW protocol would have been executed with x_2 being 2 and $P_2{}'$s sharing polynomial being $2 + 4Z$ x_3 being 0 and $P_3{}'$s sharing polynomial being $0 + 3Z$ and x_4 being 0 and $P_4{}'$s polynomial being 0 plus $3Z$ along with $P_1{}'$s input being 2. And its sharing polynomial being whatever it has used then that completely matches with all the information which actually P_1 has received during the run of the protocol.

As a result of that adversary simply cannot rule out this possibility. So, what I am trying to argue here is that even though the protocol was executed with inputs being 2, 1, 1 and 0, whatever value adversary namely P_1 has seen in that run could have also resulted if the BGW protocol would have been executed with input x_1 being 2 and input x_2 being 2 and inputs x_3 and x_4 being 0.

Now, there are other possibilities as well which along with the input x_1 being 2 can result to the sum 4. For example, one possibility could be that x_2 was 1, x_3 was 1 and x_4 was 0. Let us see whether this possibility could also result in the same set of values whether this possibility matches whatever adversary P_1 has got during the run of the protocol.

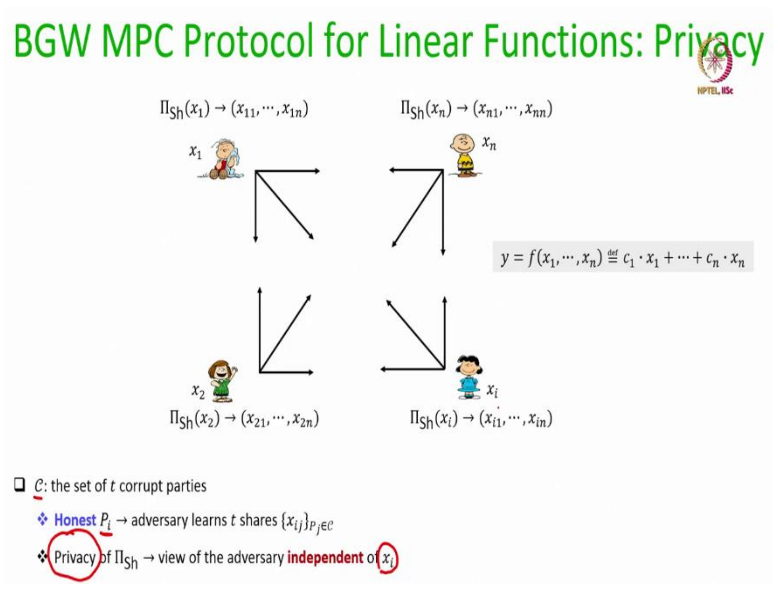
And indeed, it is quite possible that $P_2{}'$s input was 1, and its sharing polynomial was 1. $P_3{}'$s input was 1, its sharing polynomial was $1 + 2Z$, $P_4{}'$s input was 0 its sharing polynomial was $3Z$. And along with that $P_1{}'$s input was 2 and its sharing polynomial is whatever it has used. And all together it leads to a scenario where the values learnt or seen by P_1 during the run of the protocol matches.

That means adversary again cannot rule out this possibility. And in the same way adversary cannot rule out the possibility of x_2 being 0, x_3 being 2, and x_4 being 0 because indeed it could be the case that P_2 has participated in the BGW protocol with its input being 0 and sharing polynomial being Z. P_3 has participated with input being 2 and sharing polynomial being $Z + 2$. And P_4 has participated with input 0 and sharing polynomial being 3, 0.

Of course, due to lack of space in this slide, I cannot show you the other possibilities, but you can work out and you can see that magically what is happening here is that whatever is the adversary's view namely the first table which it has collected. And when I say the first table, I mean to say with this question marks here, because all those things were unknown for the adversary.

That view is going to be consistent with every candidate x_2, x_3, x_4 from the field $\mathbb{Z}_5$ such that that candidate x_2, x_3, x_4 along with x_1, x_1 being 2 leads to the value 4 and that is why adversary cannot pinpoint what exactly what the values of x_2, x_3 and x_4.

(Refer Slide Time: 13:08)



BGW MPC Protocol for Linear Functions: Privacy

$\Pi_{\text{Sh}}(x_1) \rightarrow (x_{11}, \cdots, x_{1n})$
$\Pi_{\text{Sh}}(x_n) \rightarrow (x_{n1}, \cdots, x_{nn})$
x_1
x_n
$y = f(x_1, \cdots, x_n) \overset{\text{def}}{=} c_1 \cdot x_1 + \cdots + c_n \cdot x_n$
x_2
x_i
$\Pi_{\text{Sh}}(x_2) \rightarrow (x_{21}, \cdots, x_{2n})$
$\Pi_{\text{Sh}}(x_i) \rightarrow (x_{i1}, \cdots, x_{in})$
NPTEL, IISc

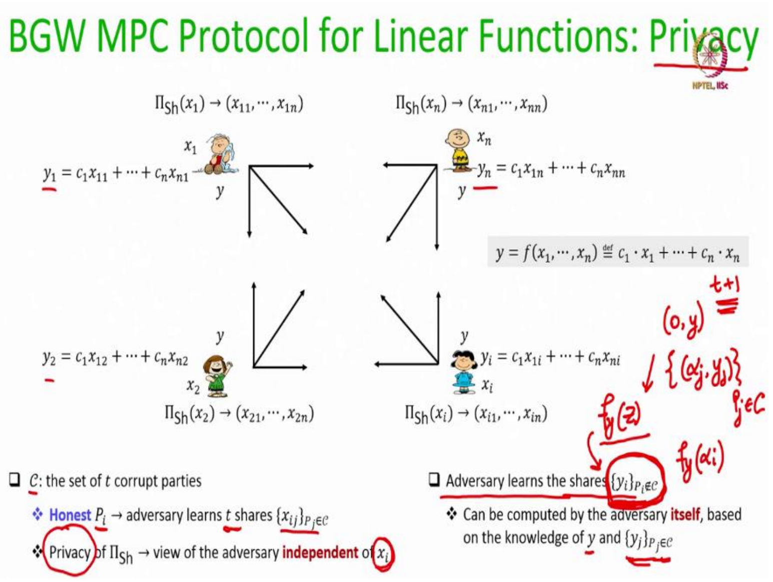
- $\mathcal{C}$: the set of t corrupt parties
  - Honest P_i → adversary learns t shares $\{x_{ij}\}_{P_j \in \mathcal{C}}$
  - Privacy of Π_{Sh} → view of the adversary **independent** of x_i

Now, let us try to understand that why this is working. For example, it is fine, but we have to give a general argument that why the adversary will fail to identify anything about the inputs of the honest parties. So, where exactly is communication happening in the protocol? The communication is happening during the input stage and during the output stage.

So, let us first fix the set of corrupt parties to the set $\mathcal{C}$. We have $2t$ corrupt parties here. So, during the sharing phase during the input stage corresponding to every honest party P_i how much information adversary learns? Adversary learns up to t shares, but we can use the privacy property of the underlying verifiable secret sharing, which guarantees that the probability distribution of the t shares which adversaries sees corresponding to the inputs of the honest parties is independent of the actual input, which is secret shared.

So, whatever t shares the adversaries sees corresponding to the inputs x_i of the honest parties that does not help the adversary to find out anything about x_i; x_i could be any random element from the field.

(Refer Slide Time: 14:24)



BGW MPC Protocol for Linear Functions: Privacy

NPTEL, IISc
$\Pi_{\mathsf{Sh}}(x_1) \rightarrow (x_{11}, \cdots, x_{1n})$
$\Pi_{\mathsf{Sh}}(x_n) \rightarrow (x_{n1}, \cdots, x_{nn})$
x_1
$y_1 = c_1 x_{11} + \cdots + c_n x_{n1}$
y
x_n
$y_n = c_1 x_{1n} + \cdots + c_n x_{nn}$
y
$y = f(x_1, \cdots, x_n) \overset{\text{def}}{=} c_1 \cdot x_1 + \cdots + c_n \cdot x_n$
$t+1$
$(0, y)$
$\{(\alpha_j, y_j)\}$
$P_j \in C$
$f_y(z)$
$f_y(\alpha_i)$
y
$y_2 = c_1 x_{12} + \cdots + c_n x_{n2}$
x_2
$\Pi_{\mathsf{Sh}}(x_2) \rightarrow (x_{21}, \cdots, x_{2n})$
y
$y_i = c_1 x_{1i} + \cdots + c_n x_{ni}$
x_i
$\Pi_{\mathsf{Sh}}(x_i) \rightarrow (x_{i1}, \cdots, x_{in})$

- ❑ $\mathcal{C}$: the set of t corrupt parties
  - ❖ **Honest** P_i → adversary learns t shares $\{x_{ij}\}_{P_j \in \mathcal{C}}$
  - ❖ Privacy of Π_{Sh} → view of the adversary **independent** of x_i
- ❑ Adversary learns the shares $\{y_i\}_{P_i \in \mathcal{C}}$
  - ❖ Can be computed by the adversary **itself**, based on the knowledge of y and $\{y_j\}_{P_j \in \mathcal{C}}$

During the computation stage, no interaction happens among the parties. So, whatever the adversary has learnt from the input stage it has learnt the same amount of information even at the end of the computation stage. And now you have the reconstruction phase or the output stage basically where the parties publicly reconstruct a function output and here adversary learns some information; learns means it receives some messages from the honest parties.

What are the messages? What are the values it receives from the honest parties? It receives basically the shares of the output value from the honest parties. Of course, the adversary has up t shares of the output value y corresponding to the corrupt parties. But now the interesting thing is that these shares corresponding to the value y received from the honest parties they are not going to add anything additional to adversary's view.

It is not going to provide any new information to the adversary because adversary already has the knowledge of the function output y because that is anyhow allowed to be learnt by everyone publicly including the adversary. And adversary itself has the t shares corresponding to t shares of the output y. To be more specific, t shares of the output y corresponding to the correct parties; that means, it has the point $(0, y)$; and it has the points (α_j, y_j) corresponding to every corrupt parties.

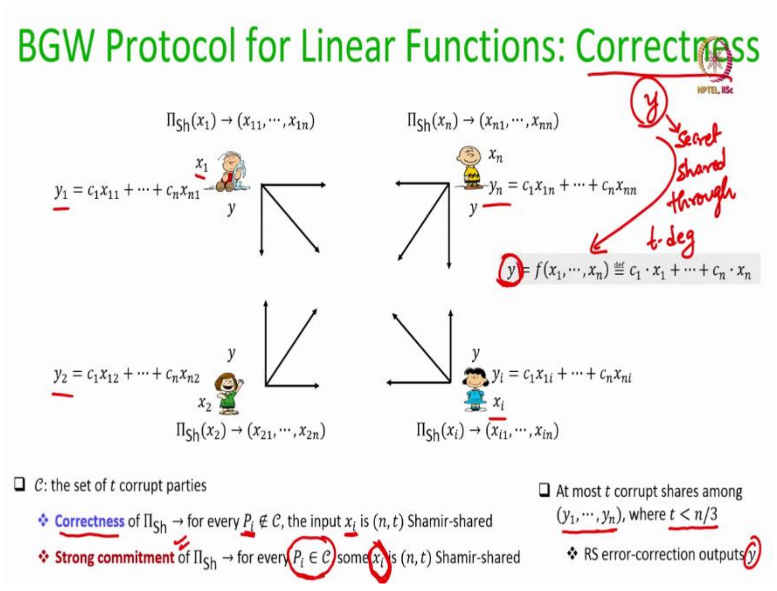
Now, using so, how many points total he has now? He has total $t + 1$ points. Now, using this $t + 1$ points, it can completely identify the output polynomial which it is going to see at the end of the Reed Solomon error correction process, because this $t + 1$ points uniquely define that polynomial which is going to be obtained at the end of the Reed Solomon error correction process call that polynomial as say $f_y(Z)$ polynomial.

Now, this $f_y(Z)$ polynomial automatically defines the y shares for the value y corresponding to the honest parties, because those shares are nothing, but the value of this f_y polynomial at α_i corresponding to every honest party P_i. And that those are the precise shares which adversary is going to receive from the honest parties during the output stage, but at the first place he already knows those shares; that means, it already knows that I am going to receive those shares from the honest parties.

So, it does not add any new information to the adversary; that means, whatever shares it is going to see it is going to receive from the honest parties as part of the output stage, it could have precomputed before itself without even waiting to see what the honest parties are sending to him. And that means, that this information this exchange of information during the output stage is nothing.

It is not going to add anything new to adversary's view and that is why we can say that adversary's view is simply limited to t shares corresponding to the inputs of the honest parties as far as the inputs of the honest parties are concerned, but anyhow the probability distribution of those shares is independent of the actual inputs secret shared by those honesty dealers and that ensures the privacy property.

(Refer Slide Time: 18:26)



BGW Protocol for Linear Functions: Correctness

$\Pi_{Sh}(x_1) \rightarrow (x_{11}, \cdots, x_{1n})$

$\Pi_{Sh}(x_n) \rightarrow (x_{n1}, \cdots, x_{nn})$

$y_1 = c_1 x_{11} + \cdots + c_n x_{n1}$

$y_n = c_1 x_{1n} + \cdots + c_n x_{nn}$

$y = f(x_1, \cdots, x_n) \overset{\text{def}}{=} c_1 \cdot x_1 + \cdots + c_n \cdot x_n$

$y_2 = c_1 x_{12} + \cdots + c_n x_{n2}$

$y_i = c_1 x_{1i} + \cdots + c_n x_{ni}$

$\Pi_{Sh}(x_2) \rightarrow (x_{21}, \cdots, x_{2n})$

$\Pi_{Sh}(x_i) \rightarrow (x_{i1}, \cdots, x_{in})$

- $\mathcal{C}$: the set of t corrupt parties
  - Correctness of Π_{Sh} → for every $P_i \notin \mathcal{C}$, the input x_i is (n, t) Shamir-shared
  - Strong commitment of Π_{Sh} → for every $P_i \in \mathcal{C}$ some x_i is (n, t) Shamir-shared
- At most t corrupt shares among $(y_1, \cdots, y_n)$, where $t < n/3$
  - RS error-correction outputs y

Now, let us see the correctness property. So, our claim here is that at the end of the protocol, every honest party will obtain the output y even if up to t corrupt parties behave maliciously. So, again let us fix the set of correct parties to $\mathcal{C}$. And during the input stage what is going to happen?

If there is an honest party P_i then even if the corrupt parties misbehave during the sharing phase protocol of those VSS instances, the correctness property guarantees that at the end of the sharing phase instances of those VSS schemes, the input x_i is (n, t) Shamir shared; that means that there will be a t degree polynomial with x_i being the constant term, and every party having a share on that polynomial.

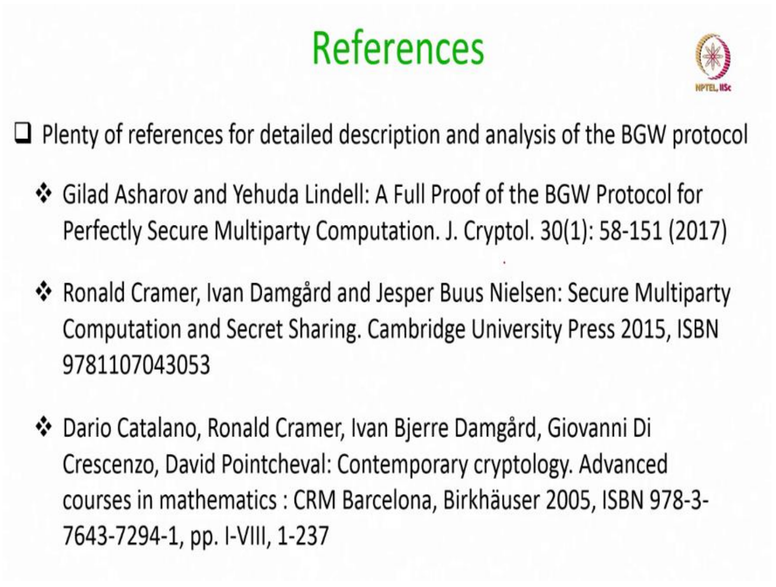
However, if the party P_i who is acting as the dealer in some secret sharing instance of the VSS scheme is corrupt. Even for such corrupt dealers the strong commitment property of the underlying secret sharing scheme guarantees that some input x_i is Shamir shared on the behalf of P_i; that means, it is not the case that corresponding to the corrupt parties no value is secret shared in an anti-Shamir shared fashion.

The strong commitment property guarantees that some value is indeed secret shared on the behalf of even potentially corrupt parties in the system; that means, at the end of the input stage all the values for respective parties are secret shared. On the behalf of the honest parties x_i will be secret shared even on the behalf of corrupt parties some values are (n, t) secret shared.

Now, the computation stage involves no interaction. So, what we can conclude is that end of the computation phase, the value y which is defined here is (n, t) secret shared. And now during the reconstruction phase, every party makes public its share of the output y up to t corrupt parties make public incorrect shares, but we are working in the setting $t < \frac{n}{3}$. And this value y is secret shared through a t degree polynomial its secret shared through t degree polynomial

So, we can now apply the properties of Reed Solomon error correction which guarantees that even there are up to t shares which are incorrect in this vector of shares for the value y. The Reed Solomon error correction algorithm will identify what those incorrect shares are and give you back the correct output y, which guarantees that even if there are up to t corrupt parties who behave maliciously in the protocol the honest parties end up obtaining the correct output y.

(Refer Slide Time: 21:43)



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

Again, my analysis and explanation here for the privacy and the correctness properties are slightly loose, because there are a bunch of other properties also which we might expect from a any generic MPC protocol. But we can prove that even those properties can also be achieved by the BGW protocol here. If you want to know more about the full rigorous analysis of the BGW protocol, you are strongly encouraged to read the first reference here.

Thank you.