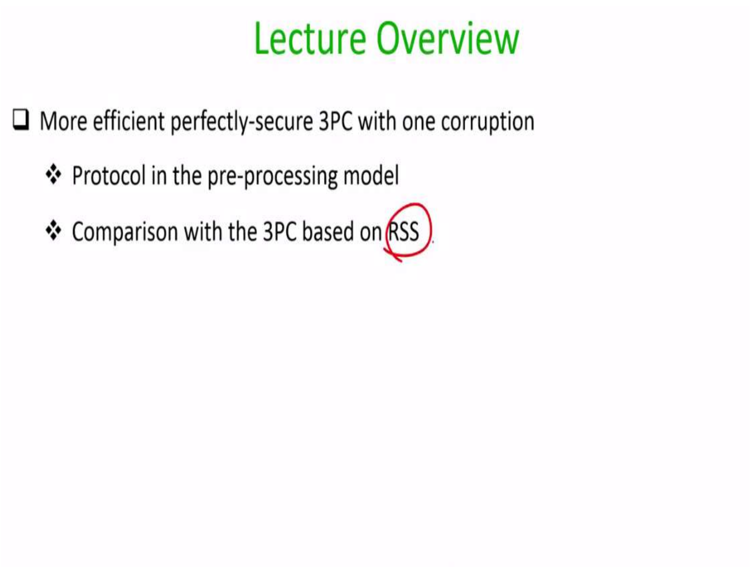
# Secure Computation: Part 1

**Prof. Ashish Choudhury**
**Department of Computer Science**
**Indian Institute of Science – Bengaluru**

## Lecture – 34
## More Efficient Perfectly-Secure 3PC Continued

Hello everyone. Welcome to this lecture.

**(Refer Slide Time: 00:31)**

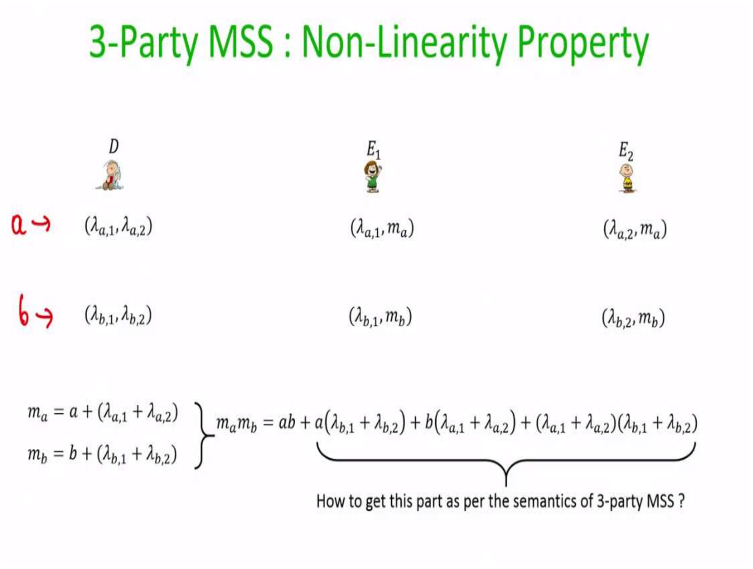


### Lecture Overview

- ❑ More efficient perfectly-secure 3PC with one corruption
  - ❖ Protocol in the pre-processing model
  - ❖ Comparison with the 3PC based on RSS.

So, in this lecture we will continue our discussion on more efficient perfectly secure 3PC based on the new secret sharing scheme that we had discussed in the last lecture and then we will finally compare our protocol with the 3PC protocol based on replicated secret sharing.

**(Refer Slide Time: 00:52)**



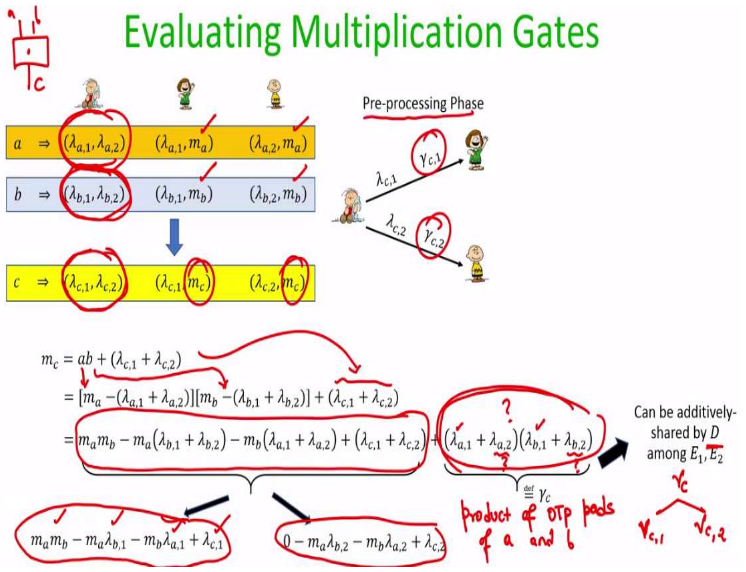
## 3-Party MSS : Non-Linearity Property

| | $D$ | $E_1$ | $E_2$ |
|---|---|---|---|
| $a \rightarrow$ | $(\lambda_{a,1}, \lambda_{a,2})$ | $(\lambda_{a,1}, m_a)$ | $(\lambda_{a,2}, m_a)$ |
| $b \rightarrow$ | $(\lambda_{b,1}, \lambda_{b,2})$ | $(\lambda_{b,1}, m_b)$ | $(\lambda_{b,2}, m_b)$ |

$$\left.\begin{array}{l} m_a = a + (\lambda_{a,1} + \lambda_{a,2}) \\ m_b = b + (\lambda_{b,1} + \lambda_{b,2}) \end{array}\right\} m_a m_b = ab + \underbrace{a(\lambda_{b,1} + \lambda_{b,2}) + b(\lambda_{a,1} + \lambda_{a,2}) + (\lambda_{a,1} + \lambda_{a,2})(\lambda_{b,1} + \lambda_{b,2})}_{\text{How to get this part as per the semantics of 3-party MSS ?}}$$

So, just to recap we discussed in the last lecture that the new secret sharing scheme that we had discussed as this non-linearity properly namely if you have values a and b which are secret shared and you want to get a times b also available in secret shared fashion then it is not possible to do that locally or non interactively we need interaction among the parties.

**(Refer Slide Time: 01:21)**



## Evaluating Multiplication Gates

| | | | |
|---|---|---|---|
| $a \Rightarrow$ | $(\lambda_{a,1}, \lambda_{a,2})$ | $(\lambda_{a,1}, m_a)$ | $(\lambda_{a,2}, m_a)$ |
| $b \Rightarrow$ | $(\lambda_{b,1}, \lambda_{b,2})$ | $(\lambda_{b,1}, m_b)$ | $(\lambda_{b,2}, m_b)$ |
| $c \Rightarrow$ | $(\lambda_{c,1}, \lambda_{c,2})$ | $(\lambda_{c,1}, m_c)$ | $(\lambda_{c,2}, m_c)$ |

Pre-processing Phase
$\lambda_{c,1}$ $\gamma_{c,1}$
$\lambda_{c,2}$ $\gamma_{c,2}$
c

$$m_c = ab + (\lambda_{c,1} + \lambda_{c,2})$$
$$= [m_a - (\lambda_{a,1} + \lambda_{a,2})][m_b - (\lambda_{b,1} + \lambda_{b,2})] + (\lambda_{c,1} + \lambda_{c,2})$$
$$= \underbrace{m_a m_b - m_a(\lambda_{b,1} + \lambda_{b,2}) - m_b(\lambda_{a,1} + \lambda_{a,2}) + (\lambda_{c,1} + \lambda_{c,2})} + \underbrace{(\lambda_{a,1} + \lambda_{a,2})(\lambda_{b,1} + \lambda_{b,2})}_{\overset{\text{def}}{=} \gamma_c}$$

$$m_a m_b - m_a\lambda_{b,1} - m_b\lambda_{a,1} + \lambda_{c,1} \qquad 0 - m_a\lambda_{b,2} - m_b\lambda_{a,2} + \lambda_{c,2}$$

Can be additively-shared by $D$ among $E_1, E_2$

Product of OTP pads of a and b

$$\begin{tikzcd} & \gamma_c \arrow[dl] \arrow[dr] & \\ \gamma_{c,1} & & \gamma_{c,2} \end{tikzcd}$$

So, here is the problem. You are given the inputs a and b so you can imagine that there is a multiplication gate whose inputs are a and b, but no one knows a no one knows b, but it is available in secret shared fashion as per the 3 party MSS sharing semantic and somehow we want to ensure that the output c which is a times b should be available in 3 party MSS sharing semantic and in the process no additional information about a and b should be disclosed.

So same problem as like your equivalent of degree reduction, but we cannot apply the degree reduction method here because the degree reduction method was applicable for the Shamir secret sharing semantics, but we are now dealing with a sharing secret sharing scheme which has a completely different sharing semantic. So, what we are going to do here? So, here is the protocol for doing this.

So, in the pre processing phase remember as per the sharing semantic the value c will have an associated OTP pad and that pad has to be available with the distributor. One of the shares of the pad with one of the evaluators another share with the other evaluator. So, those thing we can delegate to the distributor that okay you do this during the pre processing phase. In the pre processing phase the distributor will have the pads for a, pads for b and he will be knowing that okay during the circuit evaluation phase this multiplication operation is g
oing to be performed.
This multiplication gate is there which has to be evaluated by the evaluator 1 and evaluator 2. So, what it can do is for the output of that multiplication gate which is the c he will pick a uniformly random OTP pad, split it into two pieces. One of the pieces give it to the first evaluator another piece give it to the second evaluator that will be the task of the distributor and again looking ahead in the whole MPC protocol right from sharing secret sharing the inputs.

And evaluating the intermediate values the role of distributor will be only during the pre processing phase and its role will be only to associate OTP pads for all the values in the computation that is all. We are not going to involve the distributor during the evaluation of the multiplication gate. Now what do we want as part of the secret sharing of c? As part of the secret sharing of c the evaluators $E_1$ and $E_2$ should obtain the OTP encryption namely m sub c.

And that OTP encryption should be with respect to the pads $\lambda_{c1}$ $\lambda_{c2}$ that is what we want to be made available with $E_1$ and $E_2$ during the circuit evaluation phase, but we do not want to break the privacy of a and b. Now this a I can rewrite as the OTP decryption the decryption means you take the OTP encryption and unmask the mask and the mask or the pad is $\lambda_{a1}$ $\lambda_{a2}$.

Similarly b can be written as the OTP decryption namely taking out the pad from the OTP encryption and then you have the rest of the things as it is that is what is the value of OTP encryption of c. Now, I can further rewrite, rearranging the terms and bring whatever is there in the right hand side expression in this long expression and I have purposely arranged the expressions like this because now if I see here closely and focus on only this part.

This part can be rewritten, reinterpreted or this part can be interpreted as summation of two sub parts. One subpart which is computable by evaluator $E_1$ during the pre processing phase not during the pre processing phase during the circuit evaluation phase because during the circuit evaluation phase once the evaluators reach this multiplication gates where the inputs are a and b they will have the OTP encryptions of a and b available with them.

So, that is why evaluator 1 can compute the thing that I have highlighted here put in the circle. It can compute the product of the OTP encryptions of a and b and then it has the first component of the b pads so it can multiply it with the OTP encryption of a it has the first component of the a pad that it can multiply with the OTP encryption of b and in the pre processing phase it would have obtained the first component of the c pad.

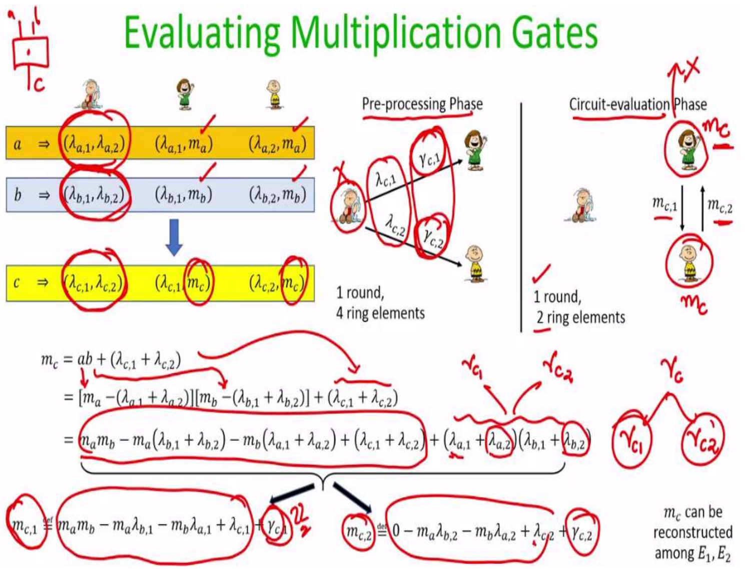
And in the same way this thing $E_2$ can compute because it has all the things now ready during the circuit evaluation phase, but then what about this part? Evaluator 1 alone cannot compute this because it has $\lambda_{a1}$, it has $\lambda_{b1}$, but it does not have $\lambda_{a2}$, it does not have $\lambda_{b2}$ and we cannot ask evaluator to that okay you go and give $\lambda_{a2}$ to evaluator 1.

And in the same way we cannot ask evaluator 1 that okay you go and give the first component of a and b pad give it to the second evaluator because then that will breach the privacy of a and b so we are kind of stuck here. But everything is not lost here because what we observe is the following. This value let me call it $\gamma_c$ and this is basically the product of OTP pads of a and b and who has this OTP pads?

Well, this OTP pads are available with the distributor because he only has picked the OTP pads with a corresponding to a and OTP pads corresponding to b and he has also picked the OTP pads corresponding to c, but this thing this ɣ c is the product of the a pad and b pad the product can be computed by the distributor and he can secret share it this value ɣ c .

So, you can imagine that ɣ c is a value he can secret share these value ɣ c by computing two random share for it say ɣ c1 ɣ c2 which adds up to ɣ c and gave it to the two evaluators respectively. So, the first piece is given to evaluator 1 the second piece is given to evaluator 2 and this can happen during the pre processing phase. Now during the circuit evaluation phase when evaluator 1 and evaluator 2 wants to compute the OTP encryption of the value c what we can do is the following. So, this was the thing right so ɣ c has been secret shared.

**(Refer Slide Time: 09:06)**



## Evaluating Multiplication Gates

$a \Rightarrow (\lambda_{a,1}, \lambda_{a,2}) \quad (\lambda_{a,1}, m_a) \quad (\lambda_{a,2}, m_a)$

$b \Rightarrow (\lambda_{b,1}, \lambda_{b,2}) \quad (\lambda_{b,1}, m_b) \quad (\lambda_{b,2}, m_b)$

$c \Rightarrow (\lambda_{c,1}, \lambda_{c,2}) \quad (\lambda_{c,1}, m_c) \quad (\lambda_{c,2}, m_c)$

Pre-processing Phase

$\lambda_{c,1}$, $\gamma_{c,1}$, $\lambda_{c,2}$, $\gamma_{c,2}$

1 round,
4 ring elements

Circuit-evaluation Phase

$m_{c,1}$, $m_{c,2}$

1 round,
2 ring elements

$$m_c = ab + (\lambda_{c,1} + \lambda_{c,2})$$
$$= [m_a - (\lambda_{a,1} + \lambda_{a,2})][m_b - (\lambda_{b,1} + \lambda_{b,2})] + (\lambda_{c,1} + \lambda_{c,2})$$
$$= \underbrace{m_a m_b - m_a(\lambda_{b,1} + \lambda_{b,2}) - m_b(\lambda_{a,1} + \lambda_{a,2}) + (\lambda_{c,1} + \lambda_{c,2}) + (\lambda_{a,1} + \lambda_{a,2})(\lambda_{b,1} + \lambda_{b,2})}$$

$$m_{c,1} \overset{\text{def}}{=} m_a m_b - m_a \lambda_{b,1} - m_b \lambda_{a,1} + \lambda_{c,1} + \gamma_{c,1} \qquad m_{c,2} \overset{\text{def}}{=} 0 - m_a \lambda_{b,2} - m_b \lambda_{a,2} + \lambda_{c,2} + \gamma_{c,2}$$

$m_c$ can be reconstructed among $E_1, E_2$

Now assuming that ɣ c has been secret shared by the distributor during the pre processing phase this value, this value has been splitted into ɣ c1, ɣ c2 and we had already seen that rest of the expressions in the OTP encryption of c it can be splitted into two parts. One part can be computed by evaluator 1 another part can be computed by evaluator 2.

So, altogether now I can say the following. Evaluator 1 can compute one of the shares for the OTP encryption of c and evaluator 2 in parallel can compute an OTP share of the OTP encryption of c and now during the circuit evaluation phase they can exchange this respective shares for the OTP encryption of c and now once they have both the shares of the OTP encryption of c they can add them and get the value m c which is what they want to do.

And now you can see that how much communication these whole thing require. So, during the pre processing phase now for this multiplication gate the distributor has to communicate four things. He has to distribute the shares of the c pad and he has to distribute the shares of the product of a and b pad. So, that is why total four things have to be communicated, but all of them can happen simultaneously because the product of the a and b pad it is independent of the c pad which the distributor is picking.

During the actual circuit evaluation the evaluators the two evaluators they have to just communicate to each other their respective shares of the OTP encryption of c that requires one round of communication and only communication of 2 ring elements. Now why this whole process preserves the privacy property? Well, of course from the view point of a potentially corrupt D if D is corrupt distributor is corrupt it does not learn anything about a and b because it is just distributing the pads associated with the c values which has got nothing to do with a and b.

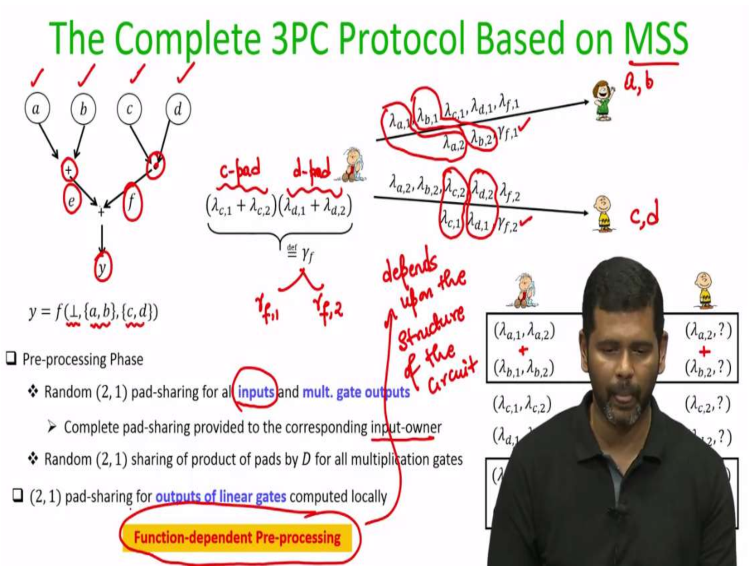
And it is distributing the product of the a and b pad again which has got nothing to do with the actual value of a and b and hence its view can be easily recreated, simulated if the distributor is correct whereas let us take the case where say the evaluator 1 is corrupt does it learn anything? So, it is learning the value $m_c$ and it is also learning the value $m_{c2}$ so since it is learning m c and it has both $m_{c1}$ and $m_{c2}$ the question is does it learn anything additional about a and b?

So, the additional thing which she can learn about a and b is whether she can learn anything about the second component of the a pad and the second component of the bad pad, but the answer is no because even though the value $\gamma_c$ has been secret shared it has been secret shared randomly in the sense when the value $\gamma_c$ has been splitted randomly into $\gamma_{c1}$, $\gamma_{c2}$.

Then basically since this second evaluator is masking the thing that he is transferring with this random $\gamma_{c2}$ it does not reveal any information to the evaluator if evaluator 1 is corrupt even though it knows $\gamma_{c1}$ because $\gamma_{c2}$ is not known to the first evaluator and running the similar symmetric argument we can argue that if say evaluator 2 is corrupt.

So it does not learn anything about whatever value evaluator 1 is sending in the sense that is padded with $\gamma_{c1}$ and $\gamma_{c1}$ is not known to the potentially corrupt $E_2$. So, it is a presence of $\gamma_{c1}$ or $\gamma_{c2}$ which preservers the privacy of information when the two evaluators are exchanging their respective shares of the OTP encryption of c to get it.

**(Refer Slide Time: 13:41)**



## The Complete 3PC Protocol Based on MSS

$y = f(\perp, \{a,b\}, \{c,d\})$

- Pre-processing Phase
  - Random (2, 1) pad-sharing for all inputs and mult. gate outputs
    - Complete pad-sharing provided to the corresponding input-owner
  - Random (2, 1) sharing of product of pads by $D$ for all multiplication gates
- (2, 1) pad-sharing for outputs of linear gates computed locally

Function-dependent Pre-processing

a
b
c
d
e
f
y
+
c-pad
d-pad
$(\lambda_{c,1} + \lambda_{c,2})(\lambda_{d,1} + \lambda_{d,2}) \overset{\text{def}}{=} \gamma_f$
$\gamma_{f,1}$
$\gamma_{f,2}$
$\lambda_{a,1}, \lambda_{b,1}, \lambda_{c,1}, \lambda_{d,1}, \lambda_{f,1}$
$\lambda_{a,2}, \lambda_{b,2}, \gamma_{f,1}$
$\lambda_{a,2}, \lambda_{b,2}, \lambda_{c,2}, \lambda_{d,2}, \lambda_{f,2}$
$\lambda_{c,1}, \lambda_{d,1}, \gamma_{f,2}$
a, b
c, d
depends upon the structure of the circuit
$(\lambda_{a,1}, \lambda_{a,2})$ + $(\lambda_{b,1}, \lambda_{b,2})$
$(\lambda_{a,2}, ?)$ + $(\lambda_{b,2}, ?)$
$(\lambda_{c,1}, \lambda_{c,2})$
$(\lambda_{c,2}, ?)$

So, now let us try to see understand the full 3PC protocol based on this new secret sharing and again I will demonstrate it. I will demonstrate it with a function where distributor has no input. So, this symbol bar denotes that distributor has no inputs. First evaluator has the inputs ab in the function and the second evaluator has the input c and d in the function. Well, again this is just for simplicity.

You can take any function and the protocol is actually secure because we know the individual secret sharing protocols or secure the linearity property ensures that the addition gates can be evaluated locally, multiplication gate we had already seen just now that during the evaluation of the multiplication gate the privacy is maintained. So, now if we stitch all this building blocks together and view the full MPC protocol it is secured.

And the view of the corrupt party can be easily simulated. So, the protocol here will consist of two phases the pre processing phase and the actual circuit evaluation phase. During the pre processing phase what the distributor is going to do is it is going to pick pads for all the values in the sense for all the circuit inputs. The inputs are here a, b, c and d and for all the multiplication gate outputs.

So, how many multiplication gates are here only one multiplication gates. So, basically it has to pick the pads $\lambda_a$ split it into two pieces distribute. So, that is a pad associated with the a input, it has to pick two random pieces for b pad and distribute it, it has to pick two random pieces for the c pad because c is one of the function inputs, it has to pick two random values as the d pad and distribute to the evaluators.

And as I said for the multiplication gate output. So, there is this multiplication gate output which has the label f. So, corresponding to that f it associates two random values which is the f pad and distribute the respective components to the two evaluators. Along with this whoever is supposed to input the corresponding input for the function the complete pad has to be available with that party because remember in the secret sharing protocol whoever wants to secret shares the value it has to own the complete pad both the components of the pad then only it can compute the full OTP encryption.

So, who are the owners of a and b here? Well, the owners of a and b are the evaluator $E_1$ and the owners of c and d is the evaluator $E_2$. So, since evaluator $E_1$ owns a and b it needs to know the full pad sharing for the a pad. So, that is why it is given both the a components. In the same way evaluator $E_1$ is the owner of the input b because it has to secret share b in the circuit evaluation phase so it has to know both the components of the b pad.

And in the same way $E_2$ is the owner of c and d values so it has to be provided with both the components of the c pad and both the components of the d pad. Apart from this during the pre processing phase remember when we are going to evaluate this multiplication gate during the circuit evaluation phase the product of the c pad and d pad needs to be available in secret shared fashion among the two evaluators.

So, that task distributor can do now itself during the pre processing. So, distributor knows here the c pad it knows the d pad. It can multiply them call the resultant value as $\gamma_f$ and $\gamma_f$ will be now randomly secret shared among evaluator 1 and evaluator 2 namely $\gamma_f$ will be splitted into two pieces two random pieces summing up to $\gamma_f$. The first piece available with the first evaluator second piece with the; second evaluator.

If there would have been other multiplication gates similar step would have been done by the distributor, but in this example circuit I am taking only one multiplication gate. So, now let us see what are the things; now that are available with the distributor evaluators. The distributor have generated the pads for a, b, c, d value and f value, it has both the components and the evaluators have only one of the components for all this pads.

The question mark here basically stands for the OTP encryptions of the a value, b value, c value, d value and so on which will be made available during the circuit evaluation phase and why this is happening because our secret sharing is asymmetric. So, the sharing semantic of our MSS secret sharing scheme is that d will have all the information associated with the secret shared value well in advance.

The evaluators will have the full information related to the full share only during the circuit evaluation phase. By the way we have not yet found the pads associated with the e value and the f value you might be wondering why distributor is not picking a random pad with the e value, random pad with the y pad and distributing. Well, we want to use the linearity property of the secret sharing.

And linearity property of the secret sharing ensures that the pads associated; with the output of the linear gate is a linear function of the pads associated with the inputs of the linear gate. So, what is the linear gate here? The plus gate that means the pad associated with the e value should be computed as the summation of the pads associated with the a and b value. So, that is why distributor is not suppose to explicitly pick a pad with the e value it will be computed once the pads with the input of the plus gate have been fixed.

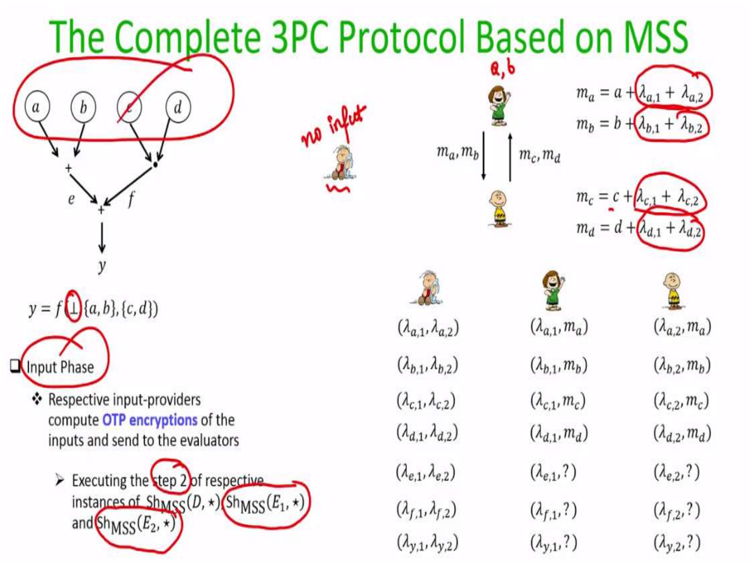
And following the same logic no explicit pad has been used with the y value because y value is the outcome of the linear gate namely the plus gate whose inputs are e value and f value and now the pads with the e value have been fixed they can be added, the pads with the f value have been fixed so it can be added with the pads associated with the e value and that will give the corresponding pads associated with the y value.

So, now if you see closely here the pre processing that is happening here that is function dependent why function dependent because it depends upon the structure of the circuit and this is unlike your previous pre processing where it has got nothing to do with what exact circuit is going to be evaluated if I consider for instance the b was pre processing we know that okay we have say m number of multiplications in the circuit.

So, just to generate m number of multiplication triples, but here you see that the pre processing that the distributor is doing it has to do something with the structure of the circuit. For instance it is associating pads only with the output of the multiplication gates, but not with the output of the linear gates because those pads have to be set as the function of the pad that you have associated with the inputs of that linear gates.

In the same way for the output of the multiplication gate for each multiplication gate the distributor has to secret share separately the product of the pads that it has associated with the inputs of the multiplication gates. So, that is why it is function dependent pre processing.

**(Refer Slide Time: 22:41)**



## The Complete 3PC Protocol Based on MSS

$m_a = a + \lambda_{a,1} + \lambda_{a,2}$

$m_b = b + \lambda_{b,1} + \lambda_{b,2}$

$m_c = c + \lambda_{c,1} + \lambda_{c,2}$

$m_d = d + \lambda_{d,1} + \lambda_{d,2}$

$y = f(\perp, \{a, b\}, \{c, d\})$

- Input Phase
  - Respective input-providers compute OTP encryptions of the inputs and send to the evaluators
    - Executing the step 2 of respective instances of $\mathsf{Sh}_{\mathsf{MSS}}(D, \star)$ $\mathsf{Sh}_{\mathsf{MSS}}(E_1, \star)$ and $\mathsf{Sh}_{\mathsf{MSS}}(E_2, \star)$

| | | |
|---|---|---|
| $(\lambda_{a,1}, \lambda_{a,2})$ | $(\lambda_{a,1}, m_a)$ | $(\lambda_{a,2}, m_a)$ |
| $(\lambda_{b,1}, \lambda_{b,2})$ | $(\lambda_{b,1}, m_b)$ | $(\lambda_{b,2}, m_b)$ |
| $(\lambda_{c,1}, \lambda_{c,2})$ | $(\lambda_{c,1}, m_c)$ | $(\lambda_{c,2}, m_c)$ |
| $(\lambda_{d,1}, \lambda_{d,2})$ | $(\lambda_{d,1}, m_d)$ | $(\lambda_{d,2}, m_d)$ |
| $(\lambda_{e,1}, \lambda_{e,2})$ | $(\lambda_{e,1}, ?)$ | $(\lambda_{e,2}, ?)$ |
| $(\lambda_{f,1}, \lambda_{f,2})$ | $(\lambda_{f,1}, ?)$ | $(\lambda_{f,2}, ?)$ |
| $(\lambda_{y,1}, \lambda_{y,2})$ | $(\lambda_{y,1}, ?)$ | $(\lambda_{y,2}, ?)$ |

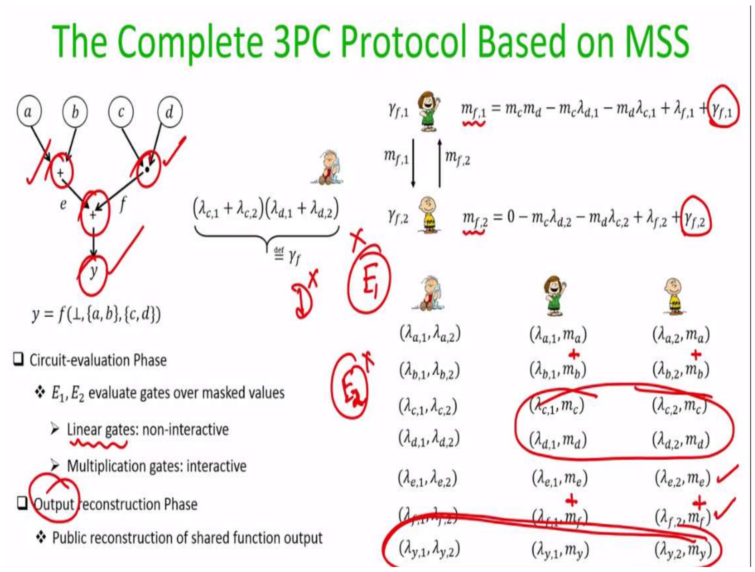
So, the pre processing is over. Now, we will come to the input phase when the parties will have the values a, b, c and d ready. They are now specific values of the a, b and c and d ready with them. So, for instance, evaluator $E_1$ now has the value of a and b. So, what it has to do? It has to basically compute the OTP encryption of and b according to the a pads and in the pre processing phase we have ensured that $E_1$ has both the a pads.

Similarly $P_1$ has been provided both the b pads so it computes the OTP encryption of the b value and gives it only to the second evaluator. So, you can imagine that this is nothing, but equivalent to executing the step 2 of all the sharing instances where $E_1$ is the dealer and in parallel the evaluator $E_2$ will compute OTP encryptions of the input c, OTP encryptions of the input d and make it available with the first evaluator.

And this is equivalent to performing the step 2 of all the sharing instance where $E_2$ is the dealer. Since we are taking an example circuit where distributor has no input that is why distributor is not supposed to compute any OTP encryption and give it to the evaluators, but if there would have been a circuit where say the distributor also own some of the inputs for the function then during the input phase the distributor once its value are ready for the circuit it will compute the OTP encryption of those respective values and give it to the two evaluators.

So, now you can see that the values a, b, c and d are completely secret shared. Till now before the input phase they were not yet completely secret shared because the OTP encryptions were not there, but once we go to the input phase and the values of a, b, c and d are decided by the input providers and they distribute the OTP encryptions the actual a, b, c and d are now available as per the sharing semantic of 3 party MSS secret sharing. So, that will be the input phase.

**(Refer Slide Time: 25:05)**



## The Complete 3PC Protocol Based on MSS

$y = f(\perp, \{a, b\}, \{c, d\})$

- Circuit-evaluation Phase
  - $E_1, E_2$ evaluate gates over masked values
    - Linear gates: non-interactive
    - Multiplication gates: interactive
- Output reconstruction Phase
  - Public reconstruction of shared function output

$$\underbrace{(\lambda_{c,1} + \lambda_{c,2})(\lambda_{d,1} + \lambda_{d,2})}_{\overset{\text{def}}{=} \gamma_f}$$

$\gamma_{f,1}$ $m_{f,1} = m_c m_d - m_c\lambda_{d,1} - m_d\lambda_{c,1} + \lambda_{f,1} + \gamma_{f,1}$

$m_{f,1} \downarrow \quad \uparrow m_{f,2}$

$\gamma_{f,2}$ $m_{f,2} = 0 - m_c\lambda_{d,2} - m_d\lambda_{c,2} + \lambda_{f,2} + \gamma_{f,2}$

| | | |
|---|---|---|
| $(\lambda_{a,1}, \lambda_{a,2})$ | $(\lambda_{a,1}, m_a)$ | $(\lambda_{a,2}, m_a)$ |
| $(\lambda_{b,1}, \lambda_{b,2})$ | $(\lambda_{b,1}, m_b)$ | $(\lambda_{b,2}, m_b)$ |
| $(\lambda_{c,1}, \lambda_{c,2})$ | $(\lambda_{c,1}, m_c)$ | $(\lambda_{c,2}, m_c)$ |
| $(\lambda_{d,1}, \lambda_{d,2})$ | $(\lambda_{d,1}, m_d)$ | $(\lambda_{d,2}, m_d)$ |
| $(\lambda_{e,1}, \lambda_{e,2})$ | $(\lambda_{e,1}, m_e)$ | $(\lambda_{e,2}, m_e)$ |
| $(\lambda_{f,1}, \lambda_{f,2})$ | $(\lambda_{f,1}, m_f)$ | $(\lambda_{f,2}, m_f)$ |
| $(\lambda_{y,1}, \lambda_{y,2})$ | $(\lambda_{y,1}, m_y)$ | $(\lambda_{y,2}, m_y)$ |

Now, we go to the circuit evaluation phase where now the evaluators 1 and 2 will start evaluating each and every gate in the circuit over the OTP encryptions and we know that for linear gates we do not require any interaction among the two evaluators and for the multiplication gates we need interaction. So, let us start with the plus gate first. For the plus gate only the evaluators have to add the OTP encryptions of and b.

And that will ensure that the value e is now available in 3 party MSS sharing semantic. So, this plus gate is evaluated then they go to the multiplication gate. Now for the multiplication gate remember in the pre processing phase the distributor would have secret shared the product of the c and d pads and we know that if the products of the c and d pads have been secret shared by the distributor.

And once this thing is available namely the OTP encryptions of the c and d which are the inputs of the multiplication gates here then the evaluator 1 can compute her share for the OTP encryption of f and that is randomized because of this and in the same way the second evaluator can compute his share for the OTP encryption of f that is randomized with this ɣf2 value and then they can exchange these respective OTP encryption shares among themselves.

And then that will ensure that the value f is secret shared as per the 3 party MSS sharing semantic and that means this multiplication gate is also evaluated. Now they go next to the plus gate. The plus gate is a linear gate whose inputs are e and f and now e and f are both ready, ready in the sense they are secret shared as per the 3 party MSS sharing semantics, the parties just need to add up parties in the sense the evaluators $E_1$ and $E_2$ just need to add up the OTP encryptions of e and f.

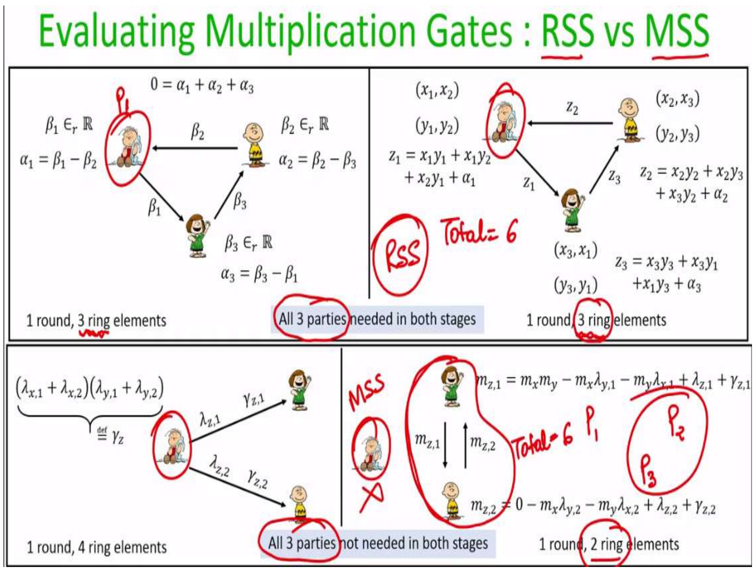
And they will get the OTP encryption of y and the circuit is evaluated completed and now there are no more gates after this. We have reached the output stage the parties can go and reconstruct publically this secret shared value using the reconstruction protocol that we had discussed in the earlier lecture and again now we can argue as per the similar argument that we have used for the BGW paradigm.

That all the values in the circuit right from the beginning to the end remained secret shared and only the output values available in the public domain and from the privacy of the secret sharing scheme the privacy of the entire computation is preserved. Of course, we can build a simulator depending upon whether the distributor is corrupt or one of the evaluators is corrupt and reproduce the entire view which that corrupt party would have obtained by participating in the this MPC protocol.

So, say for instances, if I take the case of a corrupt D then what exactly D learns in the whole MPC protocol here the 3PC protocol only the pads associated with all the values input values, multiplication gate output values, the product of the pads associated with the multiplication gate inputs and then the final y. Final y means the full sharing of the y because that is publically reconstructed.

And it is very easy to reproduce this entire view just based on the inputs of d and function output. So, same way we can simulate the entire view if even of a potentially corrupt $E_1$ if $E_1$ is corrupt similarly we can simulate entire view of a potentially corrupt $E_2$ and so on. So, I am not going into the full details you can refer to the paper that I have sighed in the last lecture otherwise you can come up with the simulator on your own it is very simple.

**(Refer Slide Time: 29:29)**


Evaluating Multiplication Gates : RSS vs MSS
$0 = \alpha_1 + \alpha_2 + \alpha_3$
$P_1$
$\beta_1 \in_r \mathbb{R}$
$\alpha_1 = \beta_1 - \beta_2$
$\beta_2$
$\beta_2 \in_r \mathbb{R}$
$\alpha_2 = \beta_2 - \beta_3$
$\beta_1$
$\beta_3$
$\beta_3 \in_r \mathbb{R}$
$\alpha_3 = \beta_3 - \beta_1$
1 round, 3 ring elements
All 3 parties needed in both stages
$(x_1, x_2)$
$(y_1, y_2)$
$z_1 = x_1y_1 + x_1y_2 + x_2y_1 + \alpha_1$
$z_2$
$(x_2, x_3)$
$(y_2, y_3)$
$z_2 = x_2y_2 + x_2y_3 + x_3y_2 + \alpha_2$
$z_1$
$z_3$
RSS
Total= 6
$(x_3, x_1)$
$(y_3, y_1)$
$z_3 = x_3y_3 + x_3y_1 + x_1y_3 + \alpha_3$
1 round, 3 ring elements
$(\lambda_{x,1} + \lambda_{x,2})(\lambda_{y,1} + \lambda_{y,2}) \overset{\text{def}}{=} \gamma_z$
$\lambda_{z,1}$
$\gamma_{z,1}$
$\lambda_{z,2}$
$\gamma_{z,2}$
1 round, 4 ring elements
All 3 parties not needed in both stages
MSS
$m_{z,1} = m_x m_y - m_x\lambda_{y,1} - m_y\lambda_{x,1} + \lambda_{z,1} + \gamma_{z,1}$
$m_{z,1}$
$m_{z,2}$
Total- 6
$P_1$
$P_2$
$P_3$
$m_{z,2} = 0 - m_x\lambda_{y,2} - m_y\lambda_{x,2} + \lambda_{z,2} + \gamma_{z,2}$
1 round, 2 ring elements

Now finally let us evaluate let us compare the procedure for evaluating multiplication gates as per the sharing semantic of replicated secret sharing and as per the sharing semantic of this MSS secret sharing. So, this was the process for RSS based evaluation of multiplication gate and why we are comparing the evaluation of multiplication gates? Because remember the cost of any generic MPC protocol is predominated by how much communication you are doing for evaluating the multiplication gates because for the input gates and for the output gate the cost are kind of fixed.

You have to secret share the input and reconstruct the outputs the cost are more or less fixed. It is only during the evaluation of the multiplication gates the parties have to interact and how much they interact, how many bids are communicated that determines overall the communication complexity of any generic MPC protocol. So, that is why we are comparing here the multiplication process of these two 3PC protocols.

So, in the RSS based 3PC protocol the pre processing phase has a cost of three ring elements. Communication wise and the circuit evaluation phase has a communication of 3 ring elements. The number of rounds; are the same and the important thing is that you need all the 3 parties to be available in both the stages. Even if one of the parties is missing here the protocol will fail in the sense that say for instance the internet connection of this party P 1 is very, very slow.

And the messages are very, very slow or it is not coming then the pre processing phase will fail and due to the same argument during the circuit evaluation phase if any of the 3 parties is not available then the protocol would not work. Now, let us compare the MSS based protocol here. So, in terms of communication we have increased the communication in the pre processing phase because earlier it was 3 ring elements now we have made it 4 ring elements.

But we have now made a saving in the actual evaluation phase. We have bought it down from 3 to 2. More importantly we do not require all the 3 parties to be present in both the stages specifically during the circuit evaluation phase it is fine if even in the distributor is not present. The evaluators only can themselves compute the circuit and finish up the entire circuit evaluation.

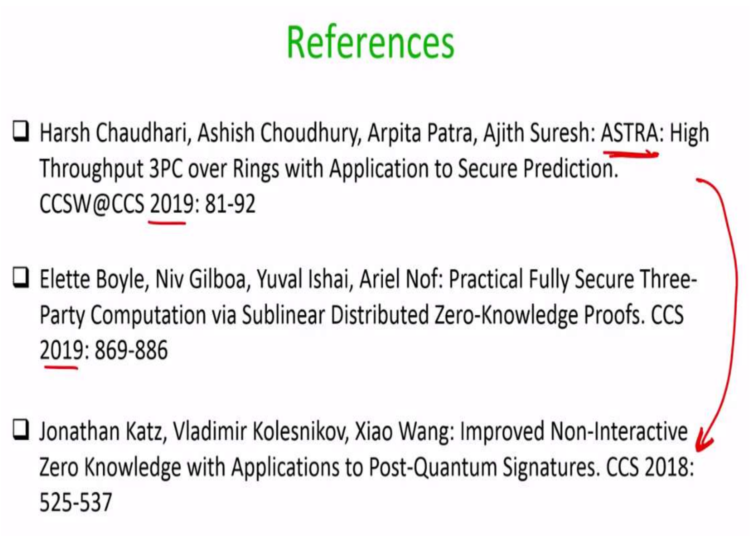
And this is a very important property because depending upon the geographical locations of the 3 parties say for instance if this party if one of the party is among $P_1$, $P_2$ and $P_3$ is very geographically kind of located remotely and his or her internet connection is kind of very slow then what we can do is we can ask that party, we can designate that party to play the role of the distributor and do whatever actions it supposed to do for the entire circuit once for all.

In one round only it can associate all the pads, it can secret share all the gamma values and then go off and then whoever are the remaining 2 parties whose internet connections are very stable, very fast they can be designated as the role of the evaluators and finish of the circuit evaluation and you do not have this kind of flexibility in the RSS based protocol because we need all the 3 parties when it comes to the shared circuit evaluation namely during the evaluation of the multiplication gate we cannot assign any special roles there.

And we have also bought down the communication complexity in this process because since we are not involving only 2 parties during the actual circuit evaluation we have bought down the communication complexity from 3 ring elements to 2 ring elements. So, even though the total communication the total is 6 ring elements in both the protocol 3 + 3 here and here also total is 4 + 2 which is 6 total is 6.

But if you focus on this important point that you do not require all the 3 parties during the actual circuit evaluation ways and during the actual circuit evaluation the communication is less in this MSS based protocol then this is a more efficient, better protocol compared to the RSS based protocol. So, with that I end today's lecture.

**(Refer Slide Time: 34:08)**



## References

- Harsh Chaudhari, Ashish Choudhury, Arpita Patra, Ajith Suresh: ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction. CCSW@CCS 2019: 81-92
- Elette Boyle, Niv Gilboa, Yuval Ishai, Ariel Nof: Practical Fully Secure Three-Party Computation via Sublinear Distributed Zero-Knowledge Proofs. CCS 2019: 869-886
- Jonathan Katz, Vladimir Kolesnikov, Xiao Wang: Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. CCS 2018: 525-537

These are again references. So, the full protocol that I had discussed today was presented in this paper as I said the similar protocol was also available actually the protocol, but in a different form was there in this 2018 paper, but we later realized it and a more efficient protocol was proposed in malicious model where the adversary could even get maliciously corrupted, but with a similar cost in this nice paper published in 2019. Thank you.