


**Secure Computation: Part II**  
**Prof. Ashish Choudhury**  
**Department of Computer Science and Engineering**  
**Indian Institute of Science, Bengaluru**

**Lecture - 47**  
**The Degree-Reduction Problem**

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:27)

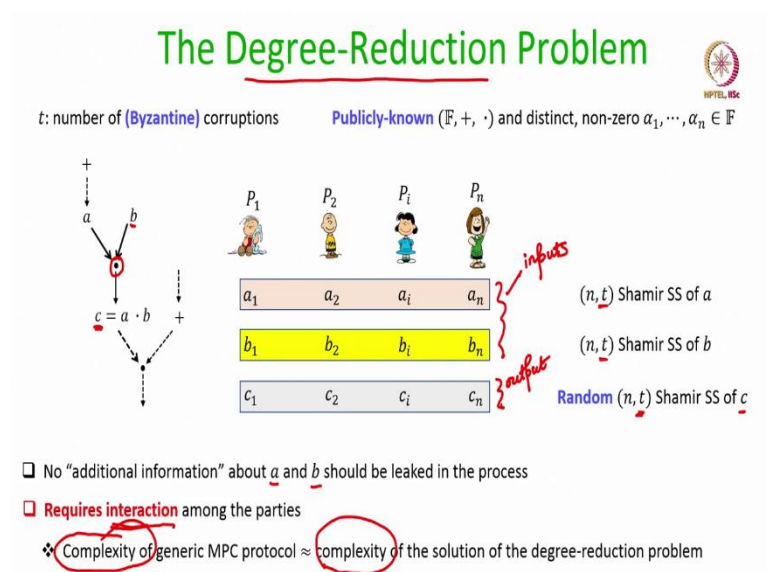
### Lecture Overview



- The degree-reduction problem in shared circuit-evaluation
  - ❖ Beaver's circuit-randomization method for the degree-reduction problem

So, in this lecture we will introduce the Degree Reduction Problem in the context of shared circuit evaluation. And, then we will discuss the Beaver's method for the degree reduction problem, Beaver's solution for the degree reduction problem.

(Refer Slide Time: 00:41)



So, this degree reduction problem occurs in the context of shared circuit evaluation in the BGW protocol. The problem is the following. Suppose you are given a circuit where we have a multiplication gate in the circuit such that the inputs of this multiplication gate, they are secret shared as per an instance of Shamir secret sharing, where the degree of the sharing is  $t$ , with each party holding its respective shares.

And, in the same way the other input here, the other gate input value  $b$  is also secret shared as per the Shamir secret sharing, where the degree of the sharing is  $t$ . And, what we want here is the following, we want a mechanism which allows us to get the gate output, let us denote it by  $c$ , where  $c$  is equal to  $a \cdot b$ . So, we require a mechanism which allows us the gate output to be randomly secret shared as per Shamir secret sharing with the degree of the sharing  $(n, t)$ .

And in the process; that means, whatever mechanism we use here to get this random secret sharing of  $c$ , it should not release any additional information about the gate inputs  $a$  and  $b$ . So that means, from the viewpoint of the adversary, adversary will know that whatever is the output of this multiplication gate that is equal to the product of the gate inputs and it will have only  $t$  shares for the gate output which are randomly distributed over the underlying field; that means, these are the inputs and this is the output.

Why is it called a degree reduction problem? Because the  $a$  and the  $b$  values they are secret shared through a  $t$  degree sharing polynomial. In the last lecture, we have seen that if every

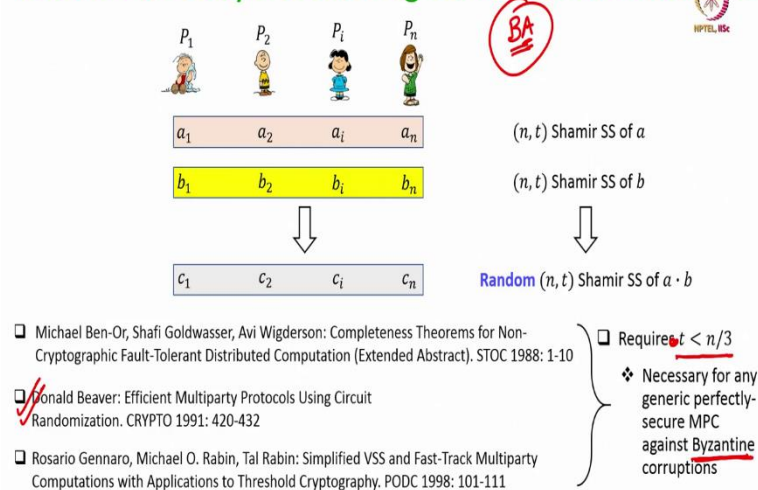
party locally multiplies its share of  $a$  and  $b$ , then that lead to secret sharing of  $c$  where the degree of the sharing will be  $2t$  and that  $2t$  degree polynomial is not a random  $2t$  degree polynomial. So, this degree reduction problem not only reduces the degree of the sharing for the  $C$  polynomial, it also ensures that the resultant  $t$  degree sharing polynomial for sharing the value  $c$  is a random  $t$  degree polynomial.

And, to solve this degree reduction problem, we require interaction among the parties. So, this degree reduction problem cannot be solved non interactively, unlike the evaluation of linear gates in the circuit.

So, it requires interaction among the parties and that is why the complexity of any generic MPC protocol is often measured in terms of the resources required to solve this degree reduction problem. That means, whatever is the complexity of this degree reduction problem, the solution for the degree reduction problem that will determine the complexity of the overall MPC protocol.

(Refer Slide Time: 04:21)

## Known Perfectly-Secure Degree-Reduction Methods

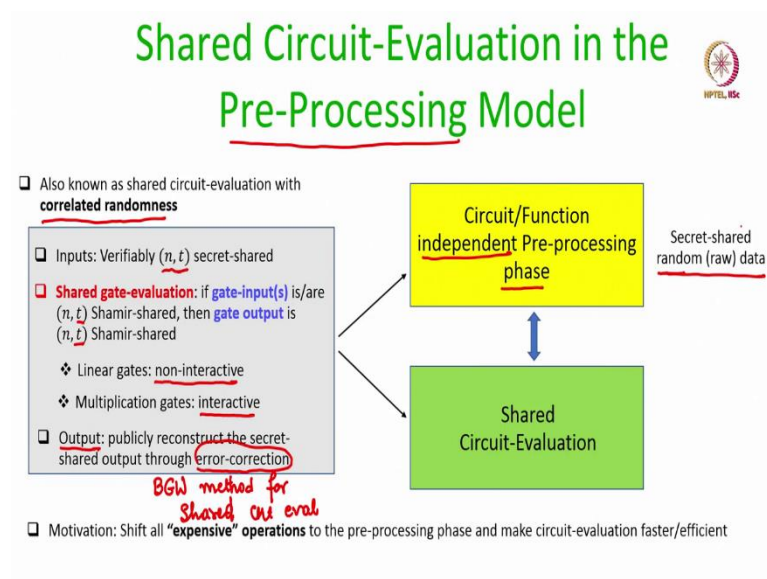


So, there are some well-known methods for solving the degree reduction problem, the original PGW paper they proposed as method for solving the degree reduction problem in the presence of  $t$  malicious adversaries. We also have a very nice method due to Don Beaver and we also have a method due to Gennaro, Rabin, Rabin. So, these are some well known methods for solving the degree reduction problem.

For our discussion, we will be following the degree reduction method attributed to Beaver. All this degree reduction methods require the condition  $t < \frac{n}{3}$  and this condition  $t < \frac{n}{3}$  is necessary for any generic perfectly secure MPC protocol tolerating up to  $t$  byzantine corruptions. So, in one of the earlier lectures I had said that this is the optimal resilience for any generic perfectly secure MPC protocols against byzantine corruptions.

Because there are certain functions which we cannot securely compute with perfect security, if this necessary condition is not met, not satisfied. For instance, the problem of byzantine agreement, the perfectly secure byzantine agreement; we cannot solve if this condition is not satisfied. And byzantine agreement is a special kind of MPC function which and this impossibility result shows that for generic functions, we require the condition  $t < \frac{n}{3}$  necessarily to get a perfectly secure method for computing the generic function.

(Refer Slide Time: 06:16)



So, now we will focus on the Beaver's method for solving the degree reduction problem. So, this method is often called as the shared circuit evaluation in the pre-processing model ok, often called as shared circuit evaluation with correlated randomness. And, the idea here is the following, this is the BGW method for shared circuit evaluation where we have 3 stages. In the input stage each party verifiably secret share its input using a polynomial based VSS. And, then we may maintain the BGW gate invariant and evaluate each gate in

the circuit in a shared fashion, where the invariant is that if the gate inputs are secret shared with the degree of sharing being  $t$ .

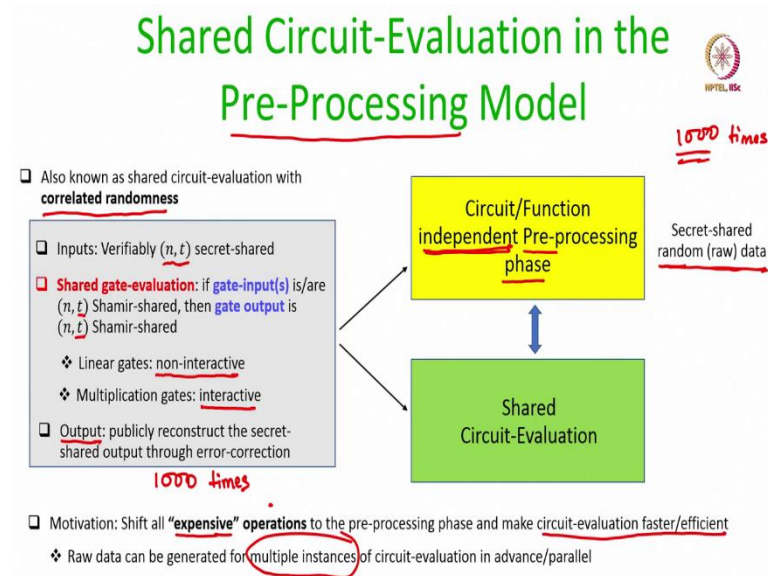
And, the gate output is also made available in a secret shared fashion where the degree of sharing is  $t$ , maintaining the invariant is free if the gate is a linear gate. For multiplication gate, we require interaction which is proportional to the complexity of the solution of the degree reduction problem. And, once we have the function output available in a secret shared fashion, the party is publicly reconstructed by exchanging the shares and if there are any incorrect shares they can be error corrected.

Now, this pre-processing model divides this I this method of shared circuit evaluation in into two independent phases. So, we have a pre-processing phase which is independent of the circuit or the function which needs to be computed, as well as the inputs of the function ok and as the name suggest this is a pre-processing phase. So, this phase can be executed well in advance before the inputs for the function are available. And, the goal for this pre-processing phase is to generate secret shared random data which has some kind of correlation among themselves.

So, no one will be knowing the value of the data, everything will be in a secret shared fashion available, and the data will be random. The point is that it will be independent of the circuit, namely the number of the namely the size of the circuit as well as the inputs for the circuit and so on. And, later when the inputs for the function are available and they are secret shared, then the parties can evaluate the circuit by utilizing the secret shared raw data, random data which they had generated in the pre processing phase.

Now, the motivation here to divide this whole process of the BGW shared circuit evaluation into two phases is the following. The motivation is that we want to shift all the computationally and communication expensive operations, namely all the operations which actually constitute the bulk of the computation and communication in the BGW protocol to the pre-processing phase, with the hope that the circuit evaluation phase becomes faster and more efficient.

(Refer Slide Time: 10:21)



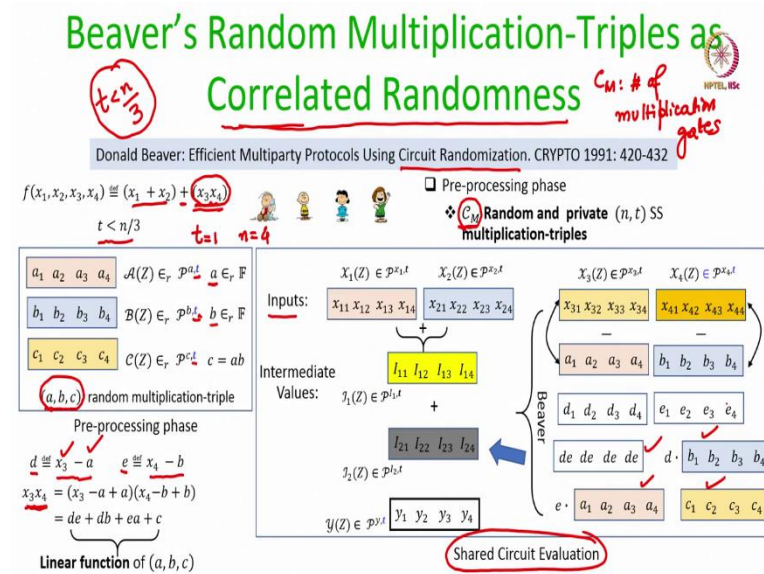
And the advantage that we will get through this process, namely this process of dividing the circuit evaluation into two phases. This process of circuit evaluation into two phases is that this raw data, it can be generated for multiple instances of the circuit evaluation in advance in parallel.

So, remember this raw data is independent of the function which needs to be computed ok. So, if the same function has to be evaluated say 1000 number of times, then in this pre-processing phase the parties can generate the raw data, the secret shared raw data which will later help them to evaluate the function  $f$  1000 number of times ok, in a faster way.

Compare it with the scenario where if you follow the original BGW method to evaluate the same circuit 1000 times, then that 1000 time circuit evaluation will happen in sequence. Because, in sequence in the sense that all the gates of the circuit will be evaluated in sequence, namely for evaluating the multiplication gates in the circuit the degree reduction problem has to be solved in sequence one after the other.

But, now what we are proposing here is a method where we could have actually preponed a lot of expensive computation in the computations in the pre-processing phase with the hope that later when the multiplication gates need to be evaluated, they can be evaluated without solving the degree reduction problem or even if we want to solve the degree reduction problem, it is not computationally expensive.

(Refer Slide Time: 12:22)



So, let us see the let us see one form of correlated randomness or the raw data which can be generated in the pre-processing phase and which significantly simplifies the evaluation of the multiplication gate or the process of solving the degree reduction problem. This method is attributed to Donald Beaver. So, this method is often called as the Beaver's circuit randomization method.

And to demonstrate the process, I take this simple circuit where I have an addition gate and a multiplication gate. We will be in the setting  $t < \frac{n}{3}$ . So, in the pre-processing phase what exactly is the raw data, that parties will generate? They will generate random secret sharing of multiplication triples, where the multiplication triples will be random for the adversary and will not be known.

Specifically, the parties will be generating  $c_M$  number of such secret shared multiplication triples, where  $c_M$  denotes the number of multiplication gates in the circuit ok. So, it is not necessary that the parties generate just  $c_M$  number of secret share multiplication triples. If it is known in advance that the same function need to be evaluated, say 1000 number of times or million number of times then and if there are  $c_M$  number of multiplication gates.

Then, they can generate 1000 times  $c_M$  or million times  $c_M$  number of secret share multiplication triples well in advance in the pre-processing phase. Now, what are multiplication triples? So, a triplet  $a, b, c$  from the field is called a multiplication triplet,

multiplication triple often Beaver triple, if  $c$  is equal to the product of  $a \cdot b$ . What do we mean by random multiplication triples?

By random multiplication triple, I mean that the  $a$  and  $b$  components are random, they are not known to the adversary. And they are randomly going to be secret shared, namely the, a component of the triplet will be secret shared through a random  $t$  degree polynomial. The  $b$  component of the triplet will be secret shared through a random  $t$  degree polynomial and the  $c$  component of the triplet is also going to be secret shared through a random  $t$  degree polynomial.

So, this is just one such secret shared random multiplication triplet, you can imagine that in the pre-processing phase the parties generate  $c_M$  number of such random secret shared multiplication triplets. How exactly they are going to generate the random secret shared multiplication triplets, that will be the focus for some of our later lectures. But right now I am going to show you that assuming that we have a pre-processing phase, where such raw data has been generated, how the degree reduction problem can be solved.

So, this will be the circuit evaluation phase, assuming that in the pre-processing phase the parties have generated the required number of secret shared multiplication triplets. So, in the circuit evaluation phase, the parties will start with the input stage where the respective input owners will verifiably secret share their respective inputs using a  $t$  degree Shamir sharing polynomial. So,  $x_1$  is secret shared here through a  $t$  degree polynomial,  $x_2$  is secret shared through a  $t$  degree polynomial,  $x_3$  is also secret shared through a  $t$  degree polynomial and  $x_4$  is secret shared through a  $t$  degree polynomial.

Here, for the purpose of demonstration, I am taking  $t = 1$  and  $n = 4$ , that constitutes the input stage. Now, the parties will start evaluating the circuit. So, they will start evaluating this plus gate. So, the inputs for this plus gate are  $x_1$  and  $x_2$  which are secret shared. So, each party can locally get its share of  $x_1 + x_2$  by simply adding its respective share of  $x_1$  and  $x_2$ .

Now, they come to the multiplication gate which requires the parties to solve an instance of the degree reduction problem. So, before going into the exact details of the Beaver's method of the degree reduction problem, solution for the degree reduction problem; let us see what exactly the underlying idea is here. So, let us define the values  $d$  and  $e$  as follows.



So,  $d$  is a masking of  $x_3$ , namely  $x_3$  being masked with  $a$  and  $e$  is  $x_4$  being masked with  $b$ . Now, if  $d$  and  $e$  are public, then it is easy to see that the product of  $x_3$  and  $x_4$  can be rewritten as  $de + db + ea + c$ . Now, this expression which is the final expression denoting the product of  $x_3$  and  $x_4$  can be interpreted as a linear function of  $a, b, c$  provided the linear combiners  $d$  and  $e$  are made public.

If the linear combiner  $d$  and  $e$  are public, then  $x_3$  and  $x_4$  can be expressed as a linear combination of  $a, b, c$ . And, since  $a, b, c$  is available in a secret shared fashion; that means, now the parties can compute a secret sharing of  $x_3 \cdot x_4$  in terms of the secret sharing of  $a, b, c$  once the combiners  $d$  and  $e$  are public. And that is precisely the idea behind the Beaver's method for solving the degree reduction problem.

So, right now  $x_3$  is secret shared,  $a$  is secret shared. So, no one can find out the value of  $d$ , but  $x_3 - a$  is a linear operation. So, parties can non-interactively compute a secret sharing of  $x_3 - a$ , namely a secret sharing of  $d$  where the degree of sharing will be  $t$ . And, in the same way the parties can non-interactively compute a secret sharing of  $e$ , where the degree of sharing will be  $t$ ; for that each party just need to seek subtract its share of  $b$  from the share of  $x_4$ .

So, now we have the secret sharing of  $d$  and the secret sharing of  $e$  and the parties can now the reconstruction protocol for publicly reconstructing  $d$  and  $e$ . So, right now  $d$  and  $e$  are in secret shared form. So, that is why they are denoted by this coloured box. So, to publicly reconstruct  $d$  each party can make public its share of  $d$  and up to  $t$  shares could be corrupt, but the degree of sharing is  $t$  and we are in the setting where  $t < \frac{n}{3}$ .

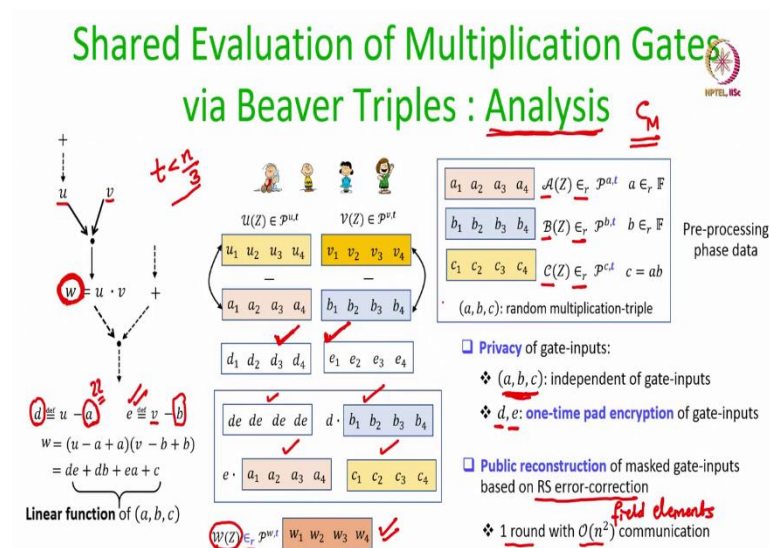
So, Reed-Solomon error correction will now work correctly and will help to recover back the value  $d$  properly, correctly. In the same way up to  $t$  shares for  $e$  could be corrupt. So, every party will be now asked to make its share of  $e$  public up to  $t$  parties can make public incorrect shares. But, since we are working with the condition  $t < \frac{n}{3}$  and  $e$  is secret shared through a  $t$  degree polynomial, the Reed-Solomon error correction will work properly and will enable the parties to reconstruct the value  $e$ .

So, now  $d$  and  $e$  are public. What is left now is to compute a secret sharing of  $x_3 \cdot x_4$  which can be now computed as a linear function of secret sharing of  $a, b, c$  because  $d$  and  $e$  are public. So, the parties can compute a secret sharing of  $d \cdot b, e \cdot a$  and anyhow the parties

have a secret sharing of  $c$  from the pre-processing phase,  $d$  and  $e$  are public. And, then by taking the linear combination, linear combination of the secret sharing they will get a secret sharing of  $x_1$ , they will get a secret sharing of  $x_3 \cdot x_4$ .

And that ensures that now we have the output of this multiplication gate available in a secret shared fashion and now the next gate is the plus gate here which is a linear gate. So, it can be evaluated non-interactively, namely each party just has to go and add its shares of this intermediate value and this intermediate value, that will help the party to get its share of the function output. So, the function output  $y$  will be now secret shared through a  $t$  degree polynomial which can be now publicly reconstructed.

(Refer Slide Time: 21:51)



So, now let us do the security analysis for the Beaver's method for solving a degree reduction problem. So, this is the idea behind getting us  $t$  degree sharing for the output value  $w$  here. The claim here is that whatever computations the parties perform in the Beaver's method, that will lead to Shamir sharing of the multiplication gate output, where the degree of the sharing will be  $t$  and that value will be secret shared through a random  $t$  degree polynomial.

So, moreover in this process the gate inputs they remain private, nothing about the gate inputs is learnt to the adversary. So, let us first prove that property that why the privacy of the gate inputs is maintained. And the privacy of the gate inputs is maintained because of the simple fact that the view of the adversary is independent of the multiplication triplet.

Namely, the adversary will not be knowing the exact value of the multiplication triplet, it will be knowing only  $t$  shares of the multiplication triplet.

So, even though, in the protocol, the adversary learns the value of  $d$  and  $e$ , because  $d$  and  $e$  are publicly reconstructed, the values  $d$  and  $e$  can be interpreted as one time pad encryptions of the gate inputs  $u$  and  $v$ . Namely,  $d$  is a masked version of the gate input  $u$  and since  $a$  is unknown for the adversary and uniformly random, the value  $d$  is also random. That means, the value  $d$  is independent of the value  $u$ , it could be any value  $u$  subtracted with any value  $a$  leading to the  $d$  which adversary learns, and the same argument holds for the value  $e$  as well.

The pad  $b$  is unknown, it is not known to the adversary. So, it could be any value  $v$  from the field subtracted with this value  $b$ , leading to the  $e$  which adversary has learnt at the protocol. So, even though the values  $d$  and  $e$  are learnt by the adversary in the protocol, they are independent of the gate inputs  $u$  and  $v$ . And, that also shows that why the parties need  $c_M$  number of multiplication triplets in the pre-processing phase. If there are  $c_M$  number of multiplication gates in the circuit, because we cannot reuse the same triplet for solving the degree reduction problem for all the multiplication gates.

For each instance of the degree reduction problem, we need a random independent triplet, that is why the need for  $c_M$  number of multiplication triplets. Now, let us argue about the correctness, namely whether the output  $w$  will be correctly secret shared through a  $t$  degree polynomial even if there are up to  $t$  malicious corruptions. So, first, note that the value  $d$  and  $e$  will be correctly reconstructed, because we are in the setting  $t < \frac{n}{3}$ .

So,  $d$  is secret shared through a  $t$  degree polynomial and when the shares of  $d$  are made public up to  $t$  shares can be made incorrectly public, up to  $t$  shares could be wrong. But the Reed-Solomon error correction guarantees that even if there are up to  $t$  incorrect shares, we have sufficient redundancy. So, that the Reed-Solomon error correction algorithm enables the parties to reconstruct the correct  $d$ , same argument holds for  $e$  as well.

Even, if up to  $t$  corrupt shares for  $e$  are made public, they can be error corrected and hence  $e$  can be reconstructed correctly. Now, once  $d$  and  $e$  are correctly reconstructed, the secret sharing of  $w$  is computed non-interactively. And it is computed non-interactively as a linear function of  $(n, t)$  secret sharing of  $b$ ;  $(n, t)$  secret sharing of  $a$  and  $(n, t)$  secret

sharing of  $c$  which guarantees that the resultant secret sharing of  $w$  is through a  $t$  degree polynomial.

Finally, why the resultant sharing of  $w$  is a random sharing? Because the resultant  $W$  polynomial is a linear combination of the secret sharing polynomials used for sharing the values  $a, b, c$  which are random right. So, remember that  $a, b, c$  is not only a random multiplication triplet, their sharings are also random; namely those values are secret shared through random  $t$  degree polynomials.

Namely, the  $A$  polynomial is a random  $t$  degree polynomial with constant term being  $a$ , the  $B$  polynomial is a random  $t$  degree polynomial with constant term being  $b$  and the  $C$  polynomial is a random  $t$  degree polynomial with the constant term being  $c$ . And, the  $W$  polynomial is a linear combination of the  $A, B$  and  $C$  polynomials which will guarantee that even the coefficients of the  $W$  polynomial are random except that its constant term will be  $w$ .

So, this shows that indeed the Beaver's method solves the degree reduction problem and achieve all the properties. It requires one round of interaction, because the only communication in the protocol is during the public reconstruction of the values  $d$  and  $e$  for which every party needs to send its share to every other party. And this will require a communication of order  $n^2$  field elements.

So, you can see that why this Beaver's method is so powerful. It solves your degree reduction problem so easily. You just need to get a random secret sharing of a multiplication triplet. And now if you want to do the degree reduction, you just require the parties to publicly reconstruct two secret shared values, where the degree of sharing is  $t$  which is very simple. After that, rest of the steps can be executed non-interact. So, with that I end this lecture.

Thank you.