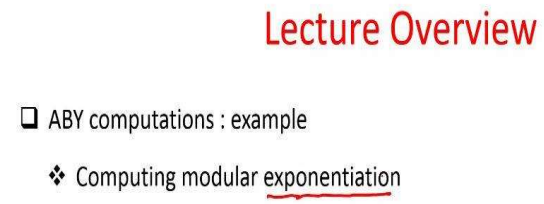
Secure Computation: Part I
Prof. Ashish Choudhury
International Institute of Information Technology, Bengaluru

Lecture - 58
ABY Computations - Example

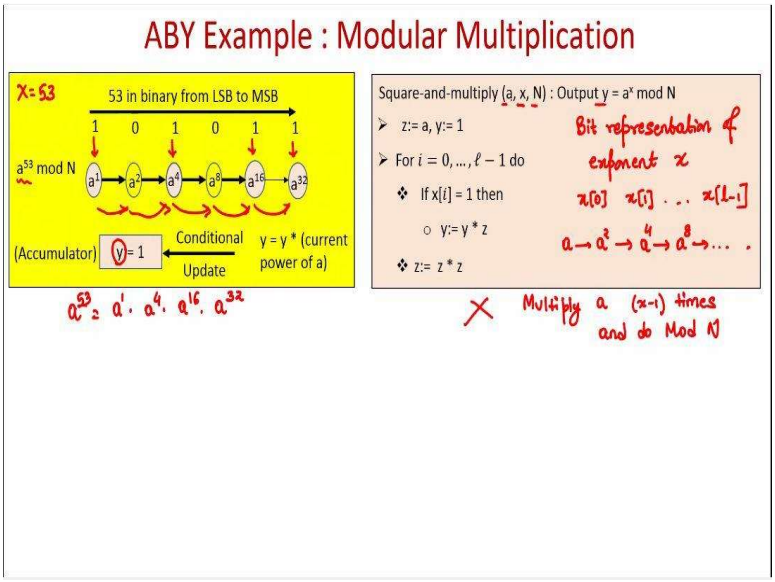
Hello everyone, welcome to this lecture. So, in this lecture, we will see an example of computation where we can do the circuit evaluation in the mixed world by using the ABY conversions.

(Refer Slide Time: 00:38)



And the computation that we will take is performing is that of performing modular exponentiation.

(Refer Slide Time: 00:49)



So, participants who are familiar with the square and multiply trick they will know that if we want to compute , where a is the base x is the exponent and it is the modulus. Then we do not do the following, we do not $a \cdot (x-1)\ \%\ N$ we do not do this because this requires exponential amount of time. Rather we perform with data we compute $a^x\ \%\ N$ by using the square and multiply trick.

And what do we do in the square and multiply trick we basically focus on the bit representation of exponent x. So, suppose the bits of x are x[0] x[1] all the way to x [i – 1] and what we do is in each step we square the current power of the base. We start with a^1 then we compute a^2 then we compute if you multiply a^2 with a^2, we get a^4 if we square it, we get a^8 we keep on doing this process step in each iteration.

And in each iteration, we also check whether the current beta of the exponent of x is 0 or 1. If it is 1, then whatever partial result we have computed and now, partial result means our eventual goal is to compute a^x. So, we will keep we will compute it step by step and accumulate the required powers of a to get z^x. So, if the current bit of x is 1 then we have to accumulate the current power of a that we are considering.

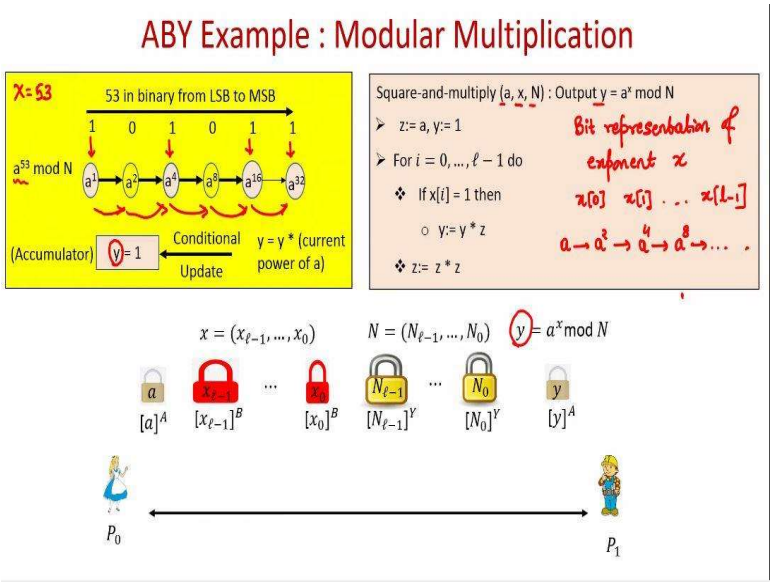
So, to demonstrate suppose, we want to compute a to the power $53\ \%\ N$ and 53 is my exponent whose bit representation from LSB to MSB stays. And a to the power 53 can be written as a$a^1 \cdot a^4 \cdot a^{16} \cdot a^{32}$. So, as I said in the in the algorithm what we do is in each iteration we go from the next power a to the from the current power f to the next higher power of a.

Where the next higher power of a is obtained by squaring the current power of a. And then we have to identify which powers of a we have to accumulate and which powers of a we have to ignore. And the powers of a which we are accumulating they will be stored in a intermediate result y and once we have accumulated all the required powers of a, we have the final result ready in y.

So, depending upon the bits of a from LSB to MSB we can identify which powers of a we have to consider which powers of a we have to ignore. Say for instance, we have the LSB of x is 1 so, we have to take a power 1 then the next bit is 0. So, ignore a square the next bit is 1. So, we have to accumulate a power 4 and so. That is a way we can compute a power x modular N. And this will not require you to perform 52 multiplications.

So, one way of computing a power 53 % N will be, you multiply itself 52 with itself 52 times. And then take modular N but instead, we can actually do, we can compute 8^{53} % N here by computing 1 2 3 4 5 and 6 7 8 9 multiplications. So, we can save exponential amount of computation here. And that is why the square and multiply trick is a very nice trick to compute a^x % N.

(Refer Slide Time: 05:44)



So now what we want to do here is the following, imagine that Alice and Bob here wants to compute a^x % N, but none of these 2 parties know the value of a value of x the value of mod N. So, the base a is arithmetic secret shared between these 2 parties, namely Alice has an l bit share, Bob has an l bit share such that if we add those l bit shares, we get the base a.

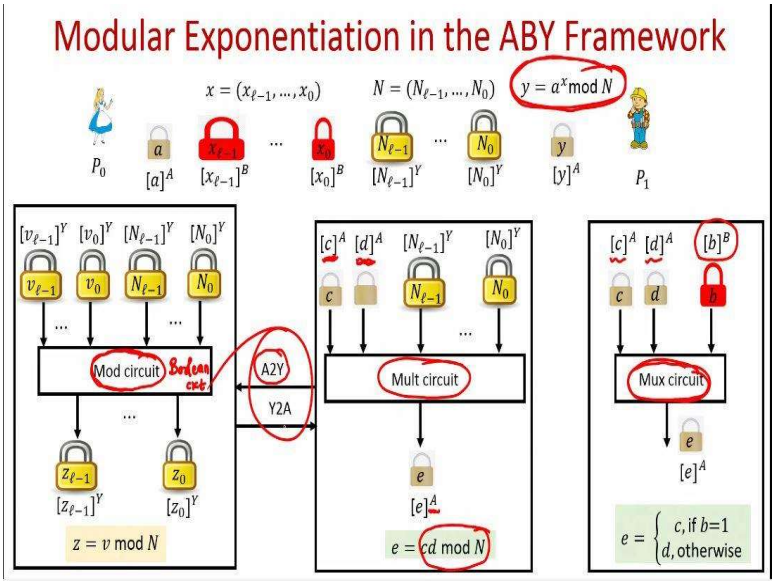
Where I say the bits of the exponent, they are Boolean secret shared between Alice and Bob. So, you can see here that we are not letting x to be arithmetic secret shared between Alice and Bob because in the square and multiply trick in each iteration Alice and Bob requires the big representation of x not the full x. So, that is why we can imagine that instead of x being secret shared between Alice and Bob bits.

The bits of x which are secret shared between Alice and Bob. And say for instance, the bits of x are secret shared between Alice and Bob as per GMW Boolean representation. Namely for each of these bits x_0 x_1 x_{i-1} Alice has a random picture Boolean share, and Bob has a random Boolean share. So, that such that if we exhort them, we get the bits of x and say the bits of N as well our secret shared between Alice and Bob, but say according to the Yao representation.

Now, our goal is to design a protocol which allows Alice and Bob to interact and let them learn the output y in a secret shared fashion where the output y is now going to be available in arithmetic representation that is what is the example computation that we are considering. So, you can imagine that a function that they are interested to compute is $a^x \% N$.

We are all the inputs a_x and N are secret shared but in different notations different representations. The base a is arithmetic secret shared bits of exponent are Boolean secret shared as per the GMW notation, and the bits of the modulus N is also secret shared as per the Yao's representation. And we want Alice and Bob to get the final output y in the arithmetic secret shared representation.

(Refer Slide Time: 08:26)



That is the goal. Now, to get this computation done, we will identify the building blocks that we will require here. So, definitely if basically we will have to evaluate, we have to run the square and multiply algorithm, but on secret shared values. In the square and multiply method, we have to do modular multiplications. Because there are two instances of multiplication one when we are computing the next power of a by squaring the current power of A.

And another conditional multiplication where we are conditionally updating the partial result. By the way, both these multiplications are modulo N, I have not written it out explicitly, but these both these multiplications are modulo N. So, we have to imagine that we have to consider a mod circuit which takes the bits of the mod N as per the Yao's representation and the value v, whose bits are represented as per the Yao representation.

And output of this circuit will be the value of the $v \% N$ as per the Yao's representation, namely the entire value v will not be secret shared but rather the entire value Z will not be secret shared, but which will be rather the bits of Z which will be secret shared as per the Yao presentation. So, this will be circuit which will be used or which will be evaluated to get this computation done.

Why so? Because as I said here, we require modular multiplications to be performed here and modular multiplications means we have to basically compute mod N. So, this $y \cdot z$ you can imagine that this is a value so, we have to compute call it as suppose y is $y \cdot z$ is 8 and we have to compute a modulo N and$z \cdot z$ is say B we have to compute be modulo N.

So, there are 2 steps where we have to perform mod of a value with respect to N. So, that is why wherever we need to do or wherever they need to compute the value of some number $v \% N$, we will invoke this mod socket. Which will ensure that neither the mod is revealed nor the value which is whose mod is computed is revealed and also the result of this computation is also not everything will be done in a secret shared fashion.

So, we can imagine that okay whatever is the mod circuit mod circuit which basically you can compute the mod $v \% N$ based on a division method or the repeated subtraction method. So, whatever is that circuit that circuits logic is here that is a Boolean circuit. And that circuit can be evaluated as per the Yao's protocol. We will also invoke multiplication circuit which will take 2 values 2 inputs in arithmetic secret shared fashion.

And the bits of mod N in the Yao representation and it produce the result e which is the product of the input c and $d \% N$. And again, everything is going to happen in secret chat fashion, but now you can see that the inputs c and d are going to be available in arithmetic secret shared fashion and the mod N will be made available in Yao secret yet fashion and output that we desire here is an arithmetic secret shared fashion.

Where exactly this mod circuit will be invoked. So, if we go here, when we are going to perform $y \cdot z$, that mult circuit will be invoked and whenever we have $z \cdot z$, we are going to invoke this mult circuit and then after that, we have $a \% N$ operation. So, to get the effect of mod N done, we will go and evaluate the mod circuit. But then that will require a switching operation, why?

Because when we perform this mult operation it will produce us the result A in the arithmetic representation and we have to then compute $e \% N$. So, e you can imagine that we will be first computing $c \cdot d$ that will be the multiplication circuit here. Now, whatever is $c \cdot d$ we have to reduce it modular N to reduce that value $(c \cdot d) \% N$ we have to take the help of this mod circuit but this mod circuit expects the input in Yao's representation.

But $(c \cdot d)$ we have computed the intermediate results $(c \cdot d)$ we have computed in the arithmetic representation that arithmetic representation has to become converted into Yao's representation and then that $(c \cdot d)$ will be reduced modulo N and then we will come back to this arithmetic part by doing Yao to arithmetic conversion and that will give us the value $(c \cdot d) \% N$.

And then we will also require a multiplexer circuit. And the logic for this multiplication circuit will be the following. It will take two inputs c and d in arithmetic representation and a bit b in the Boolean secret shared representation. And depending upon whether this selection bit b is 0 or 1 it produces either the output c or the output d in that particular presentation. So that is the logic of this mux circuit.

And where exactly this mux circuit will be invoked while compute while perform securely evaluating the square and multiply circuit. So, if the in each iteration, we check the current picked of x, if the current bit of x is 0. Then we do then we do not accumulate the current power of a whereas, if it is one then we accumulate the current power of it. So, that logic can be instantiated using a mux circuit.

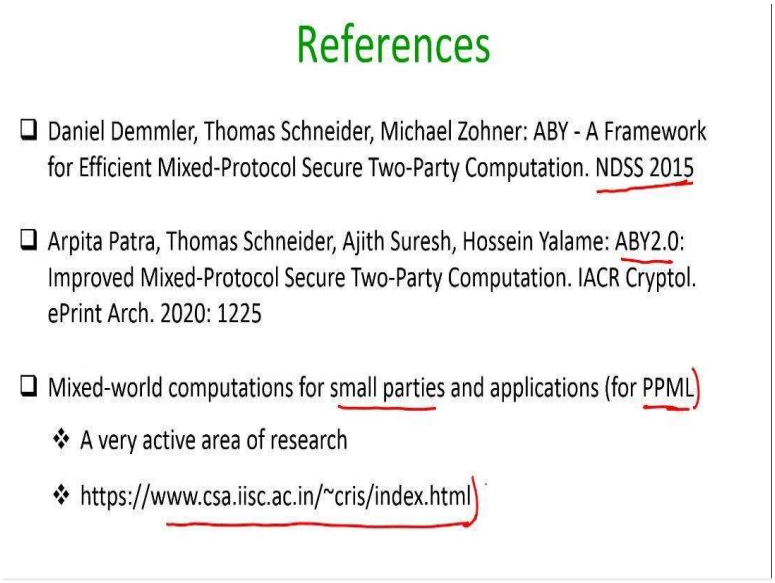
But now, the bits of x will not be available in clear but rather in secret shared fashion to model it, we are ensuring here that the selection bit which determines whether you take c is the output or d is the output is secret shared. And the inputs also which you have to select as possible outcomes are also secret shared. And now, you can imagine that by plugging in these 3 components, this mod circuit is mult circuit.

And this mux circuit, Alice and Bob can evaluate $a^x \% N$ in a secret shared fashion. In each iteration, they have to do a mult followed by mod. So, they have to do these conversions here. And then they have to evaluate a mux socket and accordingly decide whether to change the

partial result or not change the partial result. And the partial results will keep on getting accumulated.

And finally, the results will be available at us at metric secret shared fashion to LSM. Now, you can see that how the entire computation can be done in mux circuit instead of computing $a^x \% N$ entirely as per the Yao's protocol or as entirely as per the GMW protocol, we can do the different we can do different parts of the computation and different representation with the help of this switching mechanisms.

(Refer Slide Time: 16:39)



So, with that, I end the discussion on the ABY conversions. As I said, the all the discussion related to the ABY conversions are from the original paper. But recently, more efficient conversions for the two party case are proposed in this new paper. And we have the current convergence between ABY world or the three party case for party case and so. So, this makes it valid computation for a small number of parties is a very active area of research.

Because remember, we saw good practical motivation for studying makes it valid computation for a small number of parties and specifically in the context of privacy preserving machine learning. Doing machine learning computations using in a privacy preserving fashion using MPC techniques for a small number of parties is a very active area of research. You can see this link for some of the recent happenings in this area. Thank you.