


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

Lecture - 11
Dolev-Strong Reliable Broadcast Protocol: Analysis

(Refer Slide Time: 00:24)

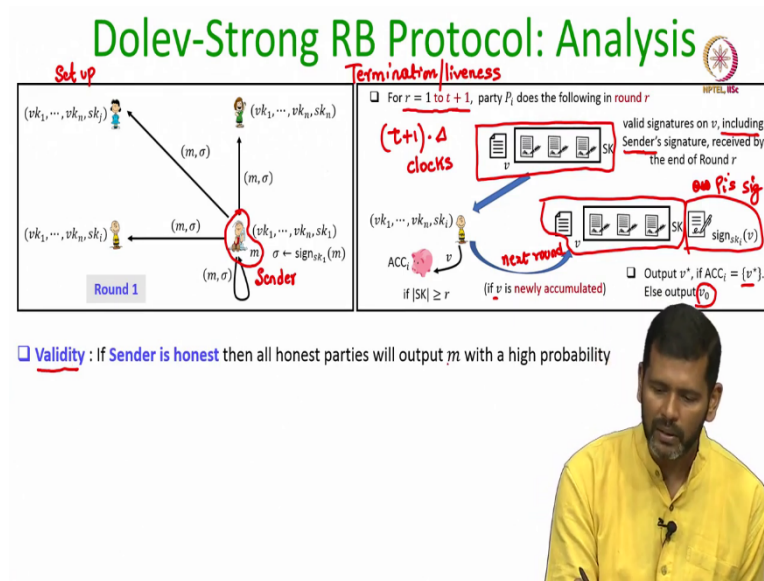
Lecture Outline



- ❑ The Dolev-Strong protocol for reliable broadcast
 - ❖ Analysis
- ❑ Cryptographically/Statistically-secure protocol for Byzantine agreement with a strict honest majority $t < \frac{n}{2}$

Hello everyone, welcome to this lecture. So, in this lecture we will do the analysis for the Dolev-Strong Protocol for Reliable Broadcast which we have discussed in the last lecture. And then using the conversion from reliable broadcast to byzantine agreement we will see a cryptographically and statistically secure version for byzantine agreement with a strict honest majority namely with the condition $t < \frac{n}{2}$.

(Refer Slide Time: 00:51)



So, let us do the analysis for the Dolev-Strong protocol; so, here is the protocol let me summarize it again for you. In the first round the sender sends its signed message to everyone. And recall that in the Dolev-Strong protocol we have this signature setup where every party has its own signing key and the verification key of all the parties will be publicly available.

So, during round one only the sender sends its signed message to everyone. And then for round r where r varies from 1 to $t + 1$ every party P_i does the following. It checks whether there are at least r signatures from r different parties available on any message any value v including the sender signature.

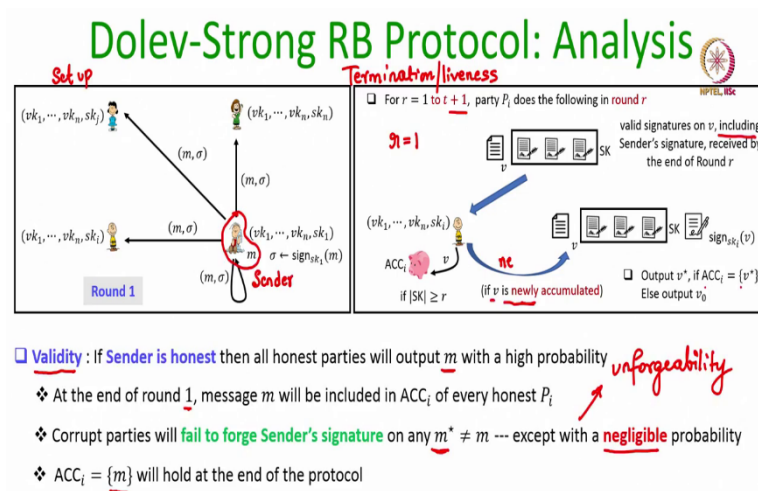
If that is the case then accumulate it if this value v is accumulated for the first time. That means, this round r is the first round during which this value v is accumulated. Then in the next round relay the collection of signatures based on which you have accumulated the value v along with your own signature P_i signature.

So, the number of signatures on the message v is now $r + 1$. So, if anyone else has not yet accumulated the value v they accumulate it in the next round. And then the decision rule for every party P_i is that, if the number of values which it has accumulated is a singleton set, namely only one value has been accumulated, then output that value otherwise output a default message. That is the protocol.

So, now we want to prove the termination, validity and consistency property, the termination or the liveness guarantee is very trivial to argue. This is because the protocol takes only $t + 1$ rounds and assuming every round takes Δ clock cycles, we know that after $(t + 1) \cdot \Delta$ clocks every party will output something. So, it will not be the case that the parties keep on running the protocol forever.

So, either they will output a default value or some value depending upon the contents of their accumulative setup; so, termination of the liveness is trivial. Now, let us prove the validity property where we want to show that if the sender is honest during the protocol execution, then all honest parties output only the sender's message with a high probability. So, let us see what happens during the protocol execution if the sender is honest. If the sender is honest then during the first round it will send its signed message m to everyone.

(Refer Slide Time: 04:33)



That means if I substitute the value of $r = 1$, then every party P_i will receive one signed m including the sender signature namely the signed m will be only from the sender itself. And hence at the end of the first round every party P_i would accumulate the message m in its accumulative set. Moreover, if there is an attacker controlling some set of parties, it might try to forge sender's signature on any other message m^* different from m and try to send to the honest parties with the hope that the signatures get accepted and honest parties accumulate

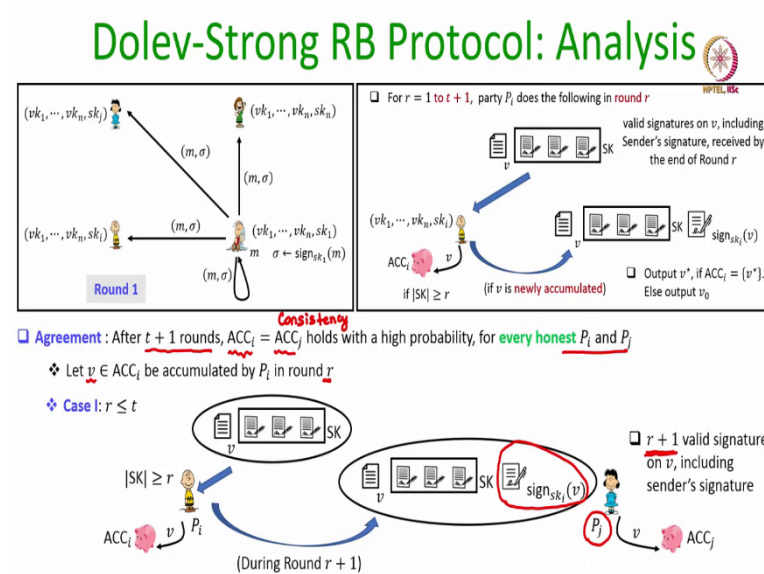
m^* . If that happens then the accumulative set of every honest party will no longer be a singleton set, it will have m and on top of that it will have m^* as well.

But what is the probability that a corrupt party can forge sender's signature on m^* different from m and spread in the system? It's very small, it is a negligible probability that comes from the unforgeability property of your signature scheme. If you are using a cryptographically secure signature, then this unforgeability property holds with the unforgeability property holds even against holds against the computationally bounded adversary.

Whereas if you are using a statistically secure, if you are using the pseudo signature scheme, then this unforgeability holds against an unbounded adversary. But whatever may be the case, it will be extremely difficult for any adversary to forge sender's signature on a message m^* different from m . As a result of that the accumulative set of every party will be a singleton set at the end of the protocol. Namely, at the end of $t + 1$ rounds the only message which will be accumulate which would have been accumulated by every honest party will be the message m only.

Because adversary can never produce a signature on m^* and its only signed m which every party will keep on forwarding to every other party during the $t + 1$ rounds. And it will be accumulated only once during the first round, after that it will never be newly accumulated. Because remember once a value has been already accumulated it is never again accumulated and relayed in the next round, we do the accumulation only for the first time; so, that shows the validity property.

(Refer Slide Time: 07:19)



Now, to prove the consistency or the agreement property we want to show that at the end of $t + 1$ rounds all the honest parties have a common output even if the sender would have been corrupt. And say the sender has sent different signed messages to different honest parties during first round. Even if a corrupt sender does behave like that, we show that after $t + 1$ rounds all honest parties will be on the same page and they will have a common output.

And to prove this we will basically show that you take any pair of honest parties (P_i, P_j) after $t + 1$ rounds the values which they have accumulated will be identical. The set of values which they have accumulated namely their respective accumulative sets ACC_i and ACC_j will be identical, that is what we will prove. If you prove that the values the set of values which they have accumulated are identical, then it implies automatically that both P_i and P_j will output the same value.

Because the decision rule is the same for both P_i and P_j , i.e., they check their respective accumulative sets, if it has more than one value output a default value, if it is a singleton set then they output that value. So, how do we prove this?

So, consider an honest party P_i and say there is a value v message v which has been accumulated by P_i . Now, that value v could have been accumulated by P_i during any of the $t + 1$ rounds; it could be during the first round, during the second round, during the $t + 1$ th round; imagine, it is accumulated during round number r .

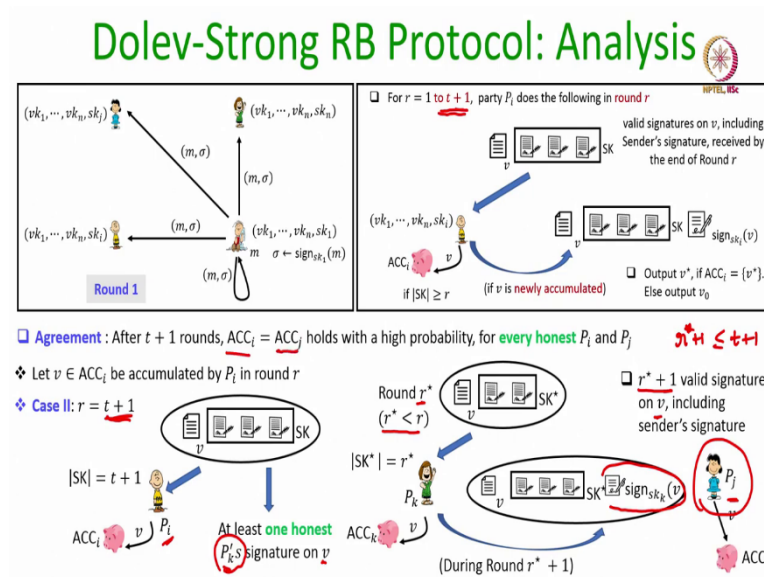
Now, why party P_i would have accumulated the value v , because it would have received certain number of signatures from parties including the sender signature on the value v . Now, there are two possible cases depending upon, when is the first round when P_i has accumulated v . So, suppose the round where P_i has accumulated v during round $r \leq t$; that means, it is within the first t rounds within which party P_i has accumulated the value v first time.

Since, it has accumulated the value v first time during the round r ; that means, it has received at least r number of signatures on the message v value v including the sender signature and it has verified them. And what it would have done, as per the protocol code there is still one more round left in the protocol because $r \leq t$; so, we still have at least $t + 1$ th round. So, in the next round namely the round $r + 1$ P_i would have endorsed the value v by relaying the r signatures that it has received on the value v along with its own signature.

And now this collection of signatures on the value v would have been received by every honest party P_j during the round $r + 1$. Now, for them it will be as if they are receiving $r + 1$ valid signatures on this message v value v including the sender signatures. And as a result, they will accumulate value v during the round $r + 1$ if they have not done that till now.

Of course, it could be possible that P_j also has accumulated the value v during the same round when P_i has accumulated it; so, that is also a possibility. But what I am saying is that if at all there is any honest party P_j who has not yet accumulated the value v by the end of round r they will do so by the end of round $r + 1$. That means, that shows anything which is present in ACC_i will be eventually present in ACC_j and that shows that ACC_i and ACC_j will be same for every pair of honest parties.

(Refer Slide Time: 11:41)



However, the tricky case to argue is when the value v is accumulated by P_i first time during the round $t + 1$. That means, till round t it has never received enough signed copies of v , but during the round $t + 1$ it has received $t + 1$ signed copies of the message v including the sender's signature. Now, we cannot run the previous argument, because there is no $t + 2$ th round where P_i would have relayed the signature.

Because the protocol is only for $t + 1$ rounds, this is not an everlasting process. This is not an infinite process where whenever I see something newly and I accumulate it and sign it and relay into the next round I do this process only for $t + 1$ number of rounds; so, the previous argument fails here. But now, let us try to analyze that since P_i has accumulated this value during the round $t + 1$ what would have happened for him?

It would have received $t + 1$ signed copies of the message v value v including the sender signature. And among those $t + 1$ signatures at least one signature will be from an honest party. Why? Because, since the signature schemes we are assuming to be unforgeable it is very unlikely for the corrupt parties to come up with signatures of even honest parties on this value v and send it to P_i no.

If at all there is any honest party P_k who is listed in this collection of signers who have signed a message v , it means P_k has signed the message v and send it to P_i during some round. It is

not the case that someone has cooked up P_k 's signature on the message v and forwarded it to P_i in this collection along with this collection of signatures on the message v that is not the case.

Now, let us see why P_k would have signed the message v and forwarded it to P_i . P_k would have accumulated the same message v during some round, because it has forwarded its own signature on the message v .

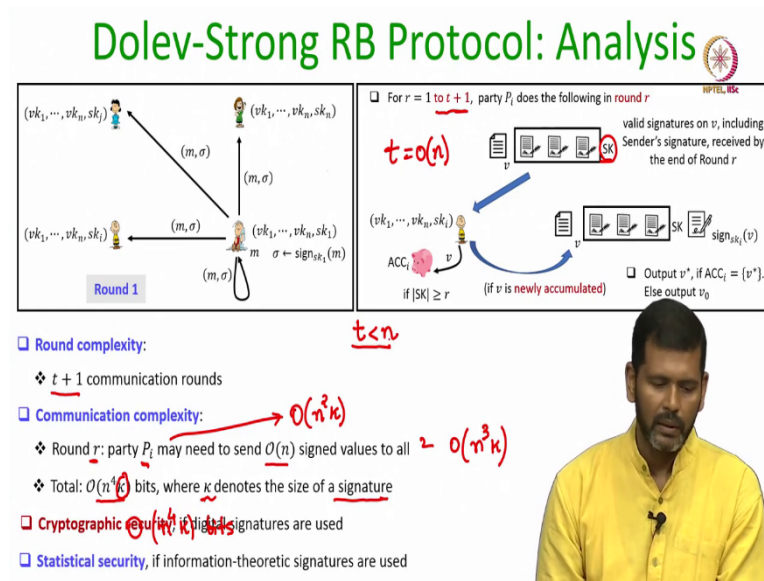
So, remember in the protocol code a party forwards its own signature on the message provided it has accumulated in some previous round. So, that previous round during which it would have accumulated that message v will be r^* and remember r^* is strictly less than r . Why? Because during the round r P_k signature would have arrived; that means, at least one round back P_k would have accumulated that value.

So, during some round r^* where r^* is strictly less than r , P_k would have accumulated the value v in its accumulative set. Namely, it would have got at least r^* number of signatures on the message v value v including the sender signature. And in the $r^* + 1$ th round it would have forwarded that collection of signatures along with its own signature which would have been received by every honest party P_j during the round $r^* + 1$; where $r^* + 1 \leq t + 1$.

So, that means, even if P_i accumulates the value v during the last round every other honest party would have accumulated that value. Because among the collection of signatures, signers among the collection of signers who have signed v at least one is guaranteed to be honest who would have signed that value because of the unforgeability property.

And since he has signed the value and forwarded it in the $t + 1$ th round; that means, at least in some previous round he himself would have accumulated that value and forwarded it in the next round which would have been then received by every other honest party. And then every other honest party would have accumulated that value. So, that shows that even in this case the accumulative set of every pair of honest parties will be the same at the end of $t + 1$ th round.

(Refer Slide Time: 16:29)



So, that is that completes the security analysis. Now, let us do the complexity analysis, how many communication rounds are required in the protocol? There are $t + 1$ communication rounds required in the protocol and that also implies the liveness property as I have already argued. The communication complexity will be $n^4 \kappa$; where, κ denotes the size of a signature. Why so?

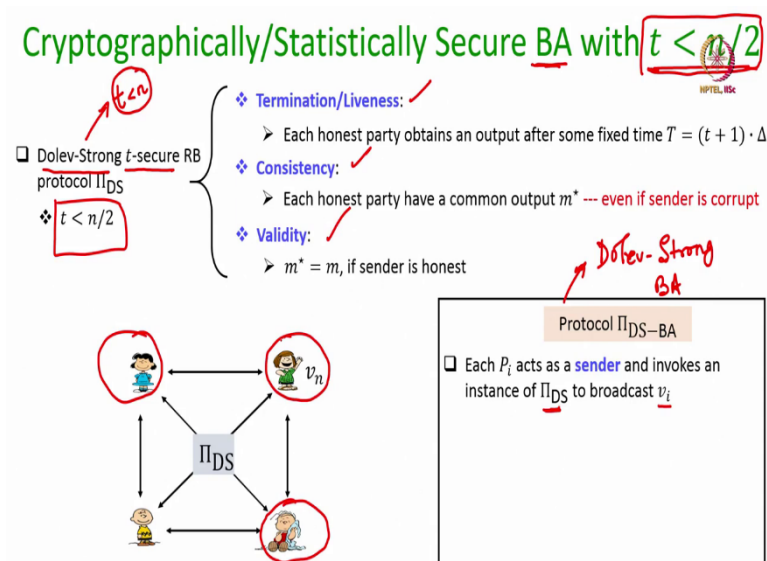
So, let us see how much communication happens in every round? In every round a party P_i may need to send a collection of $O(n)$ number of signed values to everyone else. So, remember it is forwarding a collection of signatures to every other party. So, in the worst case that collection of signatures could be as large as $O(n)$ and this it has to send to every other party. So, one single party is doing $O(n^2 \kappa)$ amount of communication.

How many such P_i 's are there? There are n such P_i 's. So, each round r involves $O(n^3 \kappa)$ amount of communication and there are total $t + 1$ rounds. So, t we can always upper bound by $O(n)$ that automatically shows the total communication in the protocol is $O(n^4 \kappa)$ bits.

And one thing I forgot to mention that the security properties namely the validity and the consistency properties that we argued are achieved even if there are $t < n$ number of corruptions. Nowhere, we use the fact that $t < \frac{n}{2}$ or $t < \frac{n}{3}$, as long as $t < n$ we are fine.

So, as I said earlier, I have explain we can the Dolev-Strong protocol is explained without assuming what kind of signature scheme is used. If the exact instantiation of the signature scheme is the crypto is with cryptographic cryptographically secure signature schemes and the resultant version of the Dolev-Strong protocol will be with cryptographic security. But if you are using information theoretic setup information theoretically secure signature scheme setup or the pseudo signatures, then the version of the Dolev-Strong protocol which you get is statistically secure.

(Refer Slide Time: 19:18)



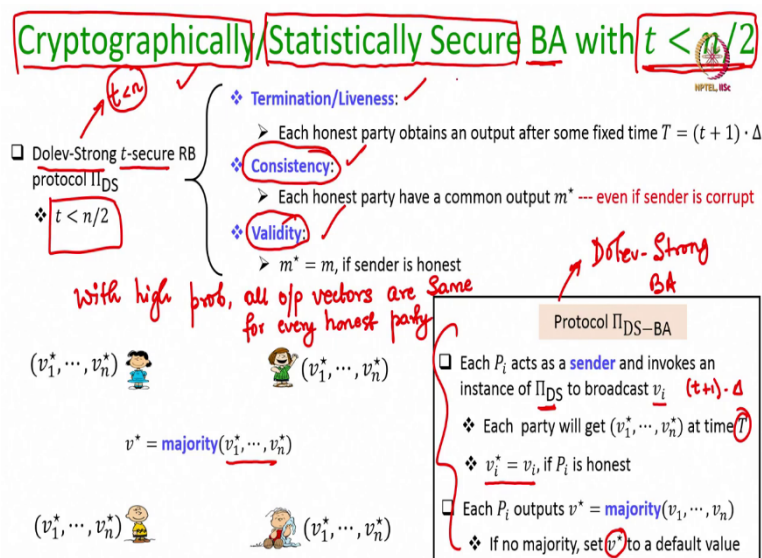
So, now let us see a cryptographically secure version of byzantine agreement with $t < \frac{n}{2}$, and the same protocol can be also considered as a statistically secure version if the underlying signature scheme is pseudo signature. So, we have already a Dolev-Strong protocol for reliable broadcast which is t secure which keeps all the security properties against t corruptions and imagine we are in the setting where $t < \frac{n}{2}$

So, the Dolev-Strong security holds even for $t < n$, but to get the byzantine agreement we are assuming now $t < \frac{n}{2}$. And this Dolev-Strong protocol it provides you termination liveness, consistency and validity against $t < \frac{n}{2}$ corruptions. Then we can use the conversion from reliable broadcast to byzantine agreement which we had seen long time back which holds where the conversion remains secure as long as $t < \frac{n}{2}$

So, that is why here I am assuming $t < \frac{n}{2}$. So, we can imagine that we have a Dolev-Strong be a protocol byzantine agreement protocol where now each party P_i acts as a sender. And invokes an instance of the Dolev-Strong protocol as a sender to broadcast its message v_i ; so, in byzantine agreement every party will have an input.

So, the protocol for the byzantine agreement here is that every party invokes an instance of the Dolev-Strong protocol as a sender to broadcast its message. Namely, $t + 1$ rounds of communication will happen for the instance of the broadcast, where P_1 is the sender with its input v_1 in parallel there will be an instance of Dolev-Strong running where P_2 is acting as the sender with input v_2 . In parallel there will be an instance of Dolev-Strong running where the i th party is the sender with input v_i . And in parallel there will be an instance of Dolev-Strong running where the n th party is the sender with input v_n .

(Refer Slide Time: 21:47)



Now, after time $(t + 1) \cdot \Delta$ which is the running time for the Dolev-Strong protocol to produce the output, every party would have received an output from all the n instances of the broadcast. So, let me denote the vector of values obtained by every party as $v_1^*, v_2^*, \dots, v_n^*$. Due to the consistency property of Dolev-Strong with high probability all output vectors are same for every honest party, all output vectors are same for every honest party.

And from the validity property of the Dolev-Strong protocol the value output value v_i^* will be same as the input v_i for the byzantine agreement which P_i has. Now, the output for the byzantine agreement is said to be the majority in the respective output vectors, if no majority is there output some default value. So, this is nothing but the conversion from the broadcast to byzantine agreement which we had seen long time back.


What we are doing here is that we are triggering that conversion in the context of the Dolev-Strong protocol and getting a variant of the byzantine agreement, which we can view as a Dolev-Strong protocol for the byzantine agreement. If the underlying signature scheme is cryptographically secure, then this byzantine agreement protocol will be cryptographically secure.

If the underlying signature scheme is information theoretically secure, then the corresponding version of byzantine agreement protocol the same protocol will be statistically secure. So, the steps of the protocol remain the same just the instantiation of the underlying signature scheme which determines whether you go for whether you get cryptographic security or whether you get statistical security.

(Refer Slide Time: 24:20)

References

- Erica Blum, Jonathan Katz, Julian Loss: Synchronous Consensus with Optimal Asynchronous Fallback Guarantees. TCC (1) 2019: 131-150



So, again for the analysis of the Dolev-Strong protocol, I have followed this reference.

Thank you.