


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

Lecture - 10
Cryptographically/Statistically-Secure Reliable Broadcast

(Refer Slide Time: 00:24)

Lecture Outline



- Cryptographically/Statistically-secure protocol for reliable broadcast with a dishonest majority $t < n$
- ❖ The Dolev-Strong protocol for reliable broadcast

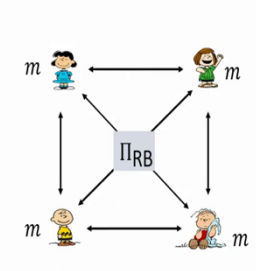
Hello everyone. Welcome to this lecture. So, till now our focus was on perfectly secure broadcast and Byzantine agreement problems. We will now shift our attention to Cryptographically Secure and Statistically Secure protocols for Byzantine agreement and Broadcast. In that context, we will see today a very popular protocol for reliable broadcast known as the Dolev-Strong protocol. And, we have two variants of this protocol; cryptographically secure as well as statistically secure, depending upon what is the type of signature scheme we are using ok.

And, this Dolev-Strong protocol works even if you have a dishonest majority; that means, it will work as long as there are t corrupt parties, where t can be as large as $n - 1$ ok. That means, the only condition which you require here is that $t < n$. This is unlike the perfectly secure protocols where you had $t < n/3$ as the necessary condition.

(Refer Slide Time: 01:32)

Reliable Broadcast (RB)

- ❑ Synchronous system, n parties, connected by pair-wise secure channels
- ❑ A designated sender party with input m
- ❑ At most t Byzantine corruptions --- **sender could be potentially corrupt**
- ❑ Security requirements of an RB protocol:
 - ❖ Termination/Liveness:
 - Each honest party obtains an output after some fixed time T
 - ❖ Consistency:
 - Each honest party should have some common output m^* --- **even if sender is corrupt**
 - ❖ Validity:
 - $m^* = m$ if sender is **honest**



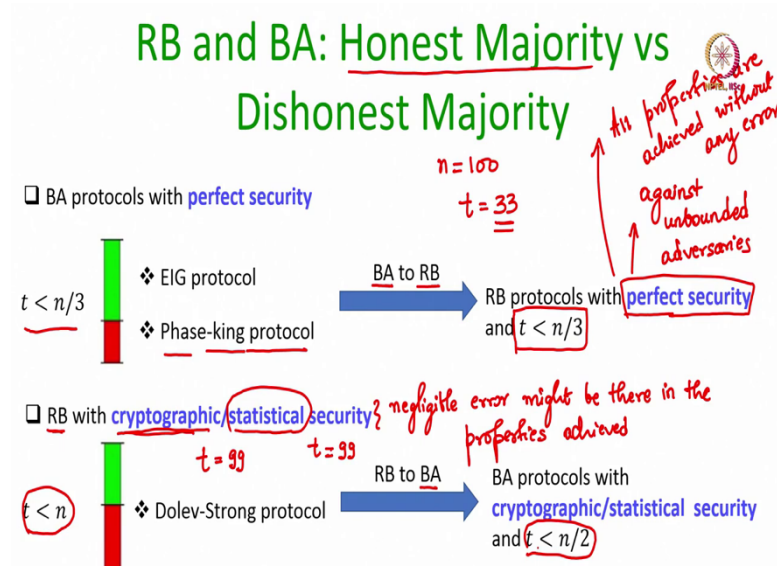
So, let me quickly recall the reliable broadcast problem RB ok. So, we have a synchronous system of n parties connected by pair-wise secure channels. And, among those n parties we have a designated sender party with some input m , from a domain it could be as simple as a single bit or it could be a message consisting of l bits. And, at most t Byzantine faults could be there in the system, potentially including the sender ok. And, we want to design a protocol according to which the parties interact with each other and obtain an output.

So, the security requirements of any reliable broadcast protocol are the following. We need the termination or the liveness guarantee which demands that every honest party should obtain an output after some fixed time, say T which will be determined by the protocol. That means, it never happens that the parties keep on running the protocol forever. We need the consistency property; that means, the output after time T for all the honest parties should be same, say m^* and this should hold even if the sender is corrupt.

That means, it should not happen that one honest party outputs a message m' and another honest party outputs message m'' and another honest party outputs a message m''' and so on. So, the consistency demands that all the honest parties should have a common output even if the sender is corrupt. That means, even if sender sends different versions of its message to different honest parties, there should be a mechanism in the protocol according to which the parties should interact and have a common output.

And, the third property is the validity property which demands that if the sender is honest during the protocol execution, then this common output m^* should be the sender's message m . It cannot be any other message different from m . So, these are the three properties which we want to achieve.

(Refer Slide Time: 03:41)



So, now let us discuss about the requirements of honest majority in the context of reliable broadcast and Byzantine agreement. So, whenever I say honest majority, I mean to say that the majority of the parties in the system are honest. And, dishonest majority means that the condition is not there; that means, more than half of the parties could be bad. So, we had seen BA protocols, Byzantine agreement protocols with perfect security where, we had $t < n/3$ as one of the necessary conditions.

So, we had seen the EIG protocol and we have seen the Phase-king protocol ok. And, we also know that the Byzantine agreement problem can be reduced to the reliable broadcast problem in the sense that, if we have a protocol for Byzantine agreement then using it we can design a protocol for reliable broadcast. So; that means, we can use the EIG protocol and the phase king protocol to even get reliable broadcast protocols with perfect security and the condition $t < n/3$.

Now, the condition $t < n/3$ was required if we want to achieve perfect security, namely security against unbounded adversaries. But, interestingly if we are fine with cryptographic

and statistical security, then we can design a reliable broadcast protocol where we just need the condition $t < n$ ok. So, it is as good it is like saying the following, say if your n is equal to 100, you have 100 participants in the protocol.

Then for perfect security you can tolerate at most 33 corruptions, even if at most 33 parties get corrupt you can achieve the required properties. And, even if those 33 parties are computationally unbounded you are fine. But, if we make the assumption that the corrupt parties are only polynomially time bounded. That means the notion of security is cryptographic security, then we can design a broadcast protocol where even if up to 99 participants are corrupt, the properties will be achieved.


And, interestingly we can achieve statistical security as well, even if the adversary is computationally unbounded we can have a statistically secure version of the Dolev-Strong protocol, where we can tolerate up to 99 corruptions. However, the notion of security will be cryptographic or statistical; that means, a negligible error might be there in the properties achieved whereas, in the perfectly secure protocols, all properties are achieved without any error ok. So, you have the trade off.

If you do not want to have any kind of compromise on the security guarantees and you want perfect security, then you have to pay a price. Namely, you can tolerate only a small fraction of corrupt parties compared to the case when you are willing to allow a very small error in the properties achieved ok.

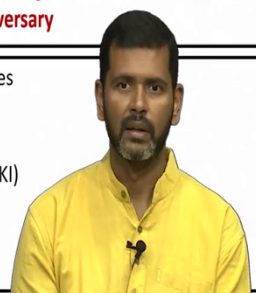
So, if we have this Dolev-Strong protocol with cryptographic security or statistical security, then using the conversion from reliable broadcast to Byzantine agreement we can in fact, get a Byzantine agreement protocol with cryptographic or statistical security as long as $t < n/2$ right.

(Refer Slide Time: 08:14)

Digital Signatures : Motivation



- ❑ Physical signatures --- tremendous real-world applications
 - ❖ Main purpose : to verify the authenticity of a document
 - Transferable: Any third party can verify the authenticity of a document
 - ❖ Security goal : forging a legitimate signature should be **difficult** for a **computationally-bounded adversary**
- ❑ Digital signatures --- digital analogue of physical signatures
 - ❖ Tremendous real-world applications
 - Digital certificates and public-key infrastructure (PKI)
 - Software updates
 - Contract signing --- legal applications



So, the Dolev-Strong protocol is based on digital signatures which is a very important cryptographic primitive. So, let us spend some time to understand what digital signatures are, what are their security properties. So, we have physical signatures which has tremendous applications in real world. The main purpose is to verify the authenticity of a document.

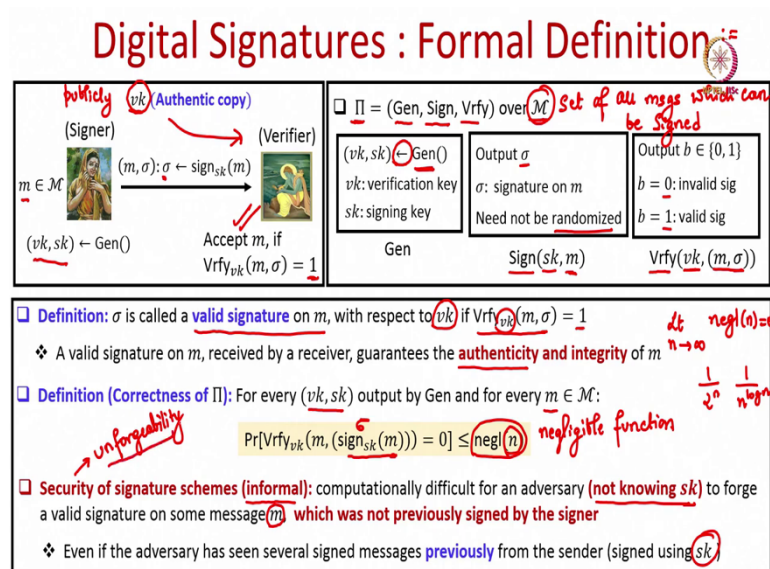
So, if I sign a document then it is authentic in the sense that anyone can show it to a third party and convince them that actually I have signed this document right, or if I sign a blank cheque right, if I have a signature on a cheque then that is an authentic piece of information, authentic piece of data for the bank, so that they can go ahead with the transaction right.

And, physical signatures are transferable in the sense that any third party can verify the authenticity of a document. So, if X have sent a signed document to Y, then Y can show it to anyone that here is a piece of document received from the party X ok. And, what is the security requirement from a physical signature? The requirement is that it should be difficult to forge a legitimate signature ok.

That means if someone has seen me signing multiple documents physically, then it should be very difficult for that person to copy my signature ok. Of course, there might be some possibility, some probability with which he can do that, but that should be very very small. So, intuitively we require same properties from digital signatures. So, digital signatures are digital versions of your physical signature. They are transferable and the main security requirement is that they should be unforgeable.

That means, it should be difficult to forge a legitimate signature for anyone who is computationally bounded. So, it has got tremendous real-world applications like digital certificates, public infrastructure, software updates, contract signing, wherever we have legal applications. In all such applications digital signatures are used on a very large scale ok.

(Refer Slide Time: 10:40)



So, let us see the formal definition of digital signatures. So, whenever we say that we have a digital signature scheme Π , we are talking about a triplet of algorithms, a key generation algorithm, a signing algorithm and a verification algorithm. And, this scheme will be over some message space, namely the set of all possible messages which can be signed using this scheme; namely, the message space which is supported by the scheme. So, it is a set of all messages which can be signed using the scheme.

Now, how this scheme will be used? So, let us first understand the syntax and semantics of each of these three algorithms, key generation, signing and verification algorithm. So, the key generation algorithm is a randomized algorithm. When I say a randomized algorithm; that means, every time you run this algorithm, it will produce different outputs with different probabilities.

So, that is why I am not using an assignment operator rather I am using this arrowhead to denote the output of the key used to express the outputs of randomized algorithms. Assignment operators are used for deterministic algorithms, to denote the outputs of the, key

generation algorithms which is a randomized algorithm, we use this arrow notation to denote the output of this key generation algorithm ok.

So, it outputs two values: a verification key and a signing key ok. So, for instance if I am the signer ok and I want to use this scheme for signing documents, then I can run the key generation algorithm myself which will output two keys for me; a verification key and a signing key. And, verification key vk will be available publicly. It will be known to everyone that this is the verification key of the so called signer.

And it will be an authentic copy; that means, it will be authenticated by some third party that indeed this verification key belongs to so and so signer. Now, the signing algorithm takes two inputs, the signing key and the message m to be signed and it outputs a signature, a bit string σ which is considered as the signature on the message. And, this algorithm, the signing algorithm need not be randomized. That means, if I sign the same message using the same signing key, its not necessary it should produce different signatures every time.

So, again if I am the signer and if I have my own signing key with me and if I want to sign a message, then what I will do is, I will run this signing algorithm which will generate a signature. And, now I can give the my message, whatever message on which I have signed, along with the signature to anyone ok. So, the both the message as well as signature is given to any party to whom I want to pass on my signed document.

And, then there will be a verification algorithm which is a two input algorithm. It takes the verification key and the message signature pair and it outputs either a value 0 or 1. 0 means the signature is considered as invalid, 1 means that the signature is considered as valid. So, again the way it is deployed is, if I have passed on a signed document m, σ to the third party, then the third party can take the verification key from the public domain and run the verification algorithm using the verification key. And, if the output of the verification algorithm is 1, then it accepts the document otherwise it rejects the document; that is typically that is typically the way digital signature scheme operates in practice. Now, a signature is called a valid signature on a message with respect to some verification key vk , if the message signature pair successfully passes the verification under the key vk ok. And, a valid signature on the message received by a receiver guarantees the authenticity and integrity of the message.

Authenticity means it is authentic, it is coming from the so called signer and integrity means that the contents have not been modified. So, we read two security properties from a signature scheme. The first is the correctness property which demands that for every pair of keys obtained by the key generation algorithm and every message m from the message space, if I sign the message using the signing key and later that message and signature are verified, then the probability that the verification fails is upper bounded by some negligible function.

So, negl here denotes a negligible function, negligible function in this parameter n , where n is the security parameter. Negligible intuitively means a function which is very very small. Its value is so small that you can consider it to be 0 for most practical purposes, if the value of n is significantly large. Namely, as limit n tends to infinity, the value of this negligible function turns to be 0. Examples of such functions 1 over 2 power n , 1 over $n \log n$, 1 over n power $\log n$ and so on ok.

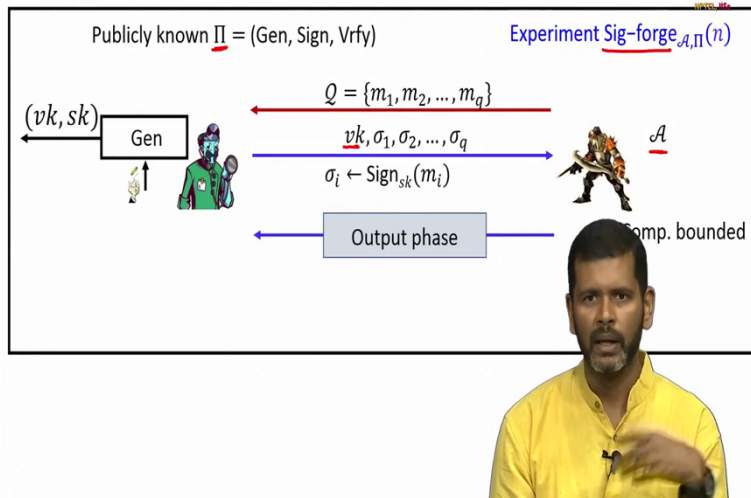
So, what does this correctness means? That means that it should never happen that if I am the signer, I obtained a pair of keys by running the key generation algorithm. And, I have legitimately signed a document and someone tries to verify that document using the corresponding verification key and the verification fails. It should never happen, the probability that it happens should be very very small.

The important property of the signature scheme is the unforgeability property ok, which demands that if there is an adversary, there is an attacker who does not know my signing key sk . Then, it should be very difficult for that adversary to produce a signature on a message which I have never signed in the past, except with some negligible probability. And, this should hold even if that attacker has seen me signing several messages in the past, using the same signing key.

And, this precisely captures the unforgeability property that we expect from the physical signature. So, if every month if I go to my bank and submit a signed cheque for some transaction and if there is a bank clerk who has seen me signing over a period of say 5 years, then still it should be difficult for that clerk to mimic my signature. Of course, it can do that, but that probability should be very small. So, intuitively we require the similar security property from a signature scheme and that is called as the unforgeability property.

(Refer Slide Time: 18:38)

Digital Signature Security : Formal Definition



Now, this unforgeability property is captured through a security experiment. So, typically in cryptography we model the security requirements through a challenge-response experiment between two entities, between an attacker and an hypothetical verifier. So, this experiment is called as the sign forge experiment. Here we have a publicly known signature scheme and there is a computationally bounded adversary.

Now, there are two phases in this experiment, a training phase and an output phase. In the training phase the adversary demands signatures of several messages of its choice. So, let Q denotes the set of messages on which adversary wants to see the signature. This basically models the fact that adversary who want to forge a signature of the sender might have seen signatures of the same sender on some earlier messages. So, those messages could be adversary's favorite messages.

So, to model that, we in this experiment provide the adversary a training where we say to the adversary that ok, you yourself get trained on messages of your choice. We will give you the signatures on those messages. So, to give the training to the adversary, the verifier will run the key generation algorithm, obtain pair of verification and signing key. And, the signing key is not revealed to the adversary, the verification key is revealed along with that signature on the messages for which adversary wants to see the signature.

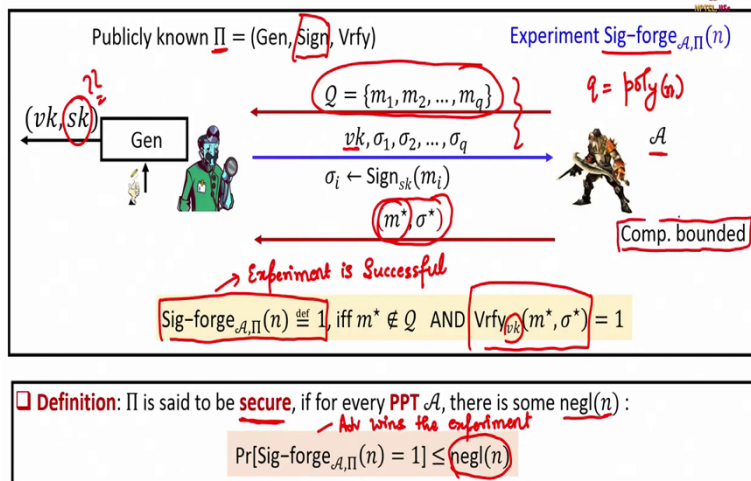
So, through this communication adversary submitting a some set of messages and getting the verification key and the signatures on those messages correspond to the real world

deployment scenario, where adversary will be knowing the verification key of the signer. It will be available in the public domain along with that adversary will have several message and signature pairs from the past which were created using the signing key, which is held by the signer.

Remember, the signing key is a very critical component. It cannot be available in the public domain, because if the signing key is available in the public domain then anyone can forge my signature ok.

(Refer Slide Time: 21:05)

Digital Signature Security : Formal Definition



So, that is a training phase and then there is an output phase where we ask the attacker that ok, let us see whether you can forge the signature or not, on any message m^* of your choice which is different from all the messages on which you have seen the signatures in the past. So, in this phase basically adversary submits a message signature pair. And, we say that the experiment is successful, which is denoted by saying that the output of the experiment is 1, if and only if the message on which adversary has produced a signature is different from all the messages on which it has seen signature in the past. And, indeed the signature σ^* is a valid signature on the message m^* under the verification key vk . This is like saying that adversary has indeed forged a signature on a new message which was not signed earlier by the sender.

We say that a signature scheme is secure, if for every polynomial time adversary who participates in this experiment there is some negligible function such that the probability


adversary wins the experiment is upper bounded by some negligible function. That means, even though there is a non-zero chance that adversary wins the experiment or adversary comes up with a forgery, that non-zero probability should be very very small. Namely, it should be a negligible function.


Why there is a possibility that adversary wins this experiment with a negligible probability? Well, adversary can simply guess my signature. It does not know the value of the signing key fine, but it knows the details of the sign algorithm. It has the message m^* on which it wants to generate a forgery. So, it can just guess a value for σ^* because, at the end of the day σ^* is a binary string. And, there is always a non-zero probability that the guessed σ^* passes the verification test.

So, that is why there is always a non-zero chance that adversary wins this experiment, but that should be very very small. And, our scheme will be called secure even if an adversary after getting trained for q number of messages, where q is some polynomial function in your security parameter, fails to forge a signature ok. There are several candidates, several instantiations for digital signature schemes ok. So, I will not be going into the details. For the Dolev-Strong protocol, we will assume that we have a digital signature scheme available.

(Refer Slide Time: 24:14)

Information-Theoretic Signature Schemes



- ❑ Also known as **pseudo-signature** schemes
 - ❖ Pfitzmann, B., Waidner, M.: Information-theoretic pseudosignatures and byzantine agreement for $t \geq n/3$. IBM (1996)
- ❑ **Unforgeability** property achieved even against a **computationally-unbounded** adversary (except with some **negligible error** probability)
- ❑ **"Limited" transferability** property 
 - ❖ Set as a parameter of the scheme

Now, in the digital signature schemes we assumed the computationally bounded adversary. Namely, it should be difficult for a computationally bounded adversary to come up with a

forgery. We have a corresponding equivalent version of the signature schemes which are called as pseudo signatures and they remain secure even against computationally unbounded adversaries.

When I say secure, I mean to say they remain unforgeable even if the adversary is computationally unbounded. That means, if we go into the experiment then even a computationally unbounded adversary who has seen signatures on several messages failed to come up with a signature on a new message except with some negligible probability ok. However, these information theoretic signature schemes which are also called as pseudo signature schemes provide limited transferability property.


That means, it will not be the case that if I sign a document and give it to you, you can show it to any number of parties, any number of third parties that is not going to be the case. This is unlike the digital signature scheme, where if I sign a document and if it is available in the public domain, it can be verified infinitely many number of times by any number of parties. Digital signature schemes where the transferability property has no restriction, namely any digitally signed document can be transferred to any number of parties, any number of third parties.


This information theoretic pseudo signatures have limited transferability property. Because, every time it gets transferred to a new party, we lose some information regarding the signing key intuitively. So, the number of times it can be transferred is set as a parameter, say P . And once a signed document is transferred P number of times, we cannot transfer it to a new third party ok. So, that way it has a limited transferability property which you can set as a parameter of the scheme right.

(Refer Slide Time: 26:24)

Dolev-Strong RB Protocol: The Set Up


- A signature-scheme setup for every party || ~~Setup~~ No Set up was required for perfectly-Secure protocols


$(vk_1, \dots, vk_n, sk_j)$


$(vk_1, \dots, vk_n, sk_n)$


❖ sk_i : signing-key for P_i

❖ vk_i : verification-key of P_i

$(vk_1, \dots, vk_n, sk_i)$


$(vk_1, \dots, vk_n, sk_1)$


- Cryptographically-secure signature scheme
 - ❖ Same set up for polynomially many instances of the protocol } Cryptographic Security
- Information-theoretic signature scheme
 - ❖ Fresh set up for each instance of the protocol } Statistical Security

So, now let us see the Dolev-Strong reliable broadcast protocol ok. And, this reliable broadcast protocol will require a setup ok, a setup for a signature scheme. So, what is the setup? Here we assume that every party has a signing key and a verification key, where the signing key for the i th party sk_i is available only to the i th party. So, here we have party 1 with its signing key, party i with its signing key, party j with its own signing key and party n with its own signing key. And, the verification key of all the parties will be known to everyone else.

So, this is the setup which we assume is present at the beginning of the protocol right. Only in the presence of this setup, we can run the Dolev-Strong protocol. So, no setup was required for perfectly secure protocols which we had seen till now right. Only setup was pair wise secure channels among the parties, but for the Dolev-Strong protocol we need actually this special setup, namely the signature scheme setup. You might be wondering who does this setup. So, this setup we can assume is done by a trusted party at the beginning of the protocol.

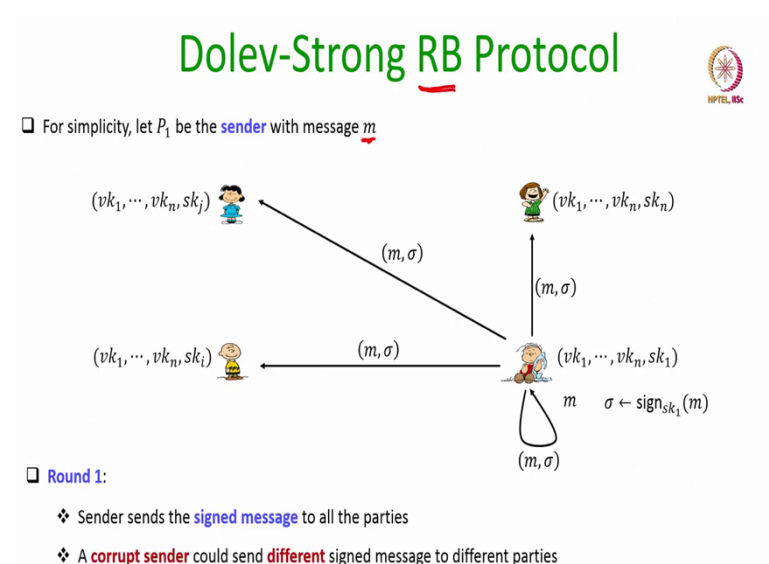
If we are using a cryptographically signature secure signature scheme, then that setup can be reused for polynomially many instances of the Dolev-Strong protocol ok. That means, say for instance a setup is used for 1000 number of instances of Dolev-Strong protocol or 100 numbers or 10,000 number of instances of Dolev-Strong protocol depending upon the exact values of your security parameter.

Whereas, if you are using the pseudo signature setup, namely the setup for the information theoretic signature scheme, then we cannot reuse the same setup for multiple invocations of the reliable or Dolev-Strong protocol. Namely, for each invocation or each instance of the RB protocol, a fresh setup has to be established; that is a downside if you are using an information theoretic signature scheme setup.

But, security wise we get more security, namely we get statistical security which will hold even against an unbounded adversary. Whereas, if you are reusing the same setup for polynomially many instances of the protocol, then we get cryptographic security ok. So, we will explain the steps of the Dolev-Strong protocol irrespective of the type of the signature scheme, whether it is cryptographically secure or whether it is information theoretic secure.

We will just assume it has a signature setup, depending upon the nature of the setup whether it is cryptographically secure or information theoretic secure, we get a cryptographically secure version of Dolev-Strong protocol or a statistically secure version of the Dolev-Strong protocol.

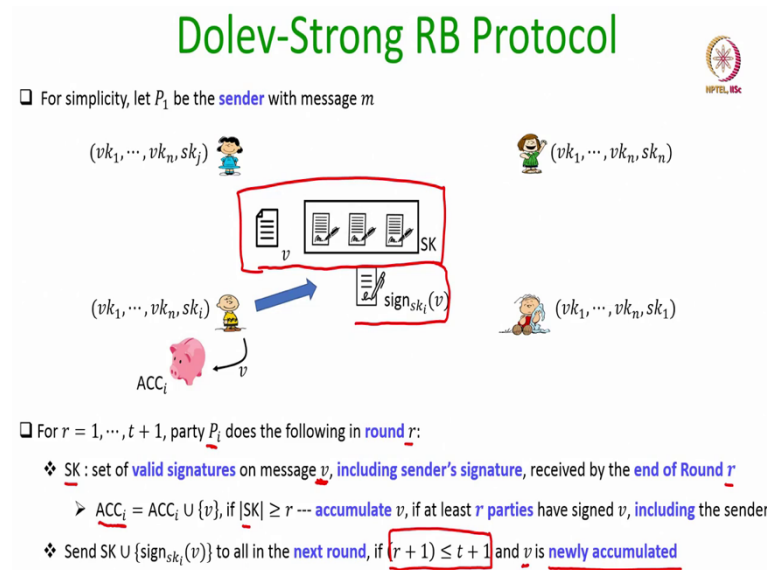
(Refer Slide Time: 30:12)



So, what are the protocol steps? So, remember in the reliable broadcast protocol we have a designated sender. So, for simplicity we assume that P_1 is the sender with some message m and protocol works if it is any sender. The assumption regarding the sender is without loss of generality. In the first round, the sender sends its signed message to everyone ok.

As a basic step, it sends the message to everyone, but to prove that it is an authentic data it is coming from the sender, it signs it and send to everyone including itself. Of course, if the sender is corrupt it can send different versions of signed message to different parties ok. But if the sender is honest, it will follow this protocol instruction correctly and it will send an identical copy of the signed message to everyone.

(Refer Slide Time: 31:09)



Now, the rest of the protocol, the remaining rounds of the protocol will be interaction among the parties to identify whether sender has sent different signed messages to different parties ok. Namely, for round r equal to 1 to $t + 1$, every party P_i does the following in round r . So, it maintains a set of signed documents, namely it maintains a set of valid signatures on any message v , including sender signature which has been received till now namely by the end of round r .

So, by the end of round r party P_i might have received several signed messages. Among all those signed messages whichever are the valid signed messages, it maintains them in a set SK . The only criteria is that among those signed messages for any value v or any message v , there should be sender's signature also which is present there; otherwise it will not be included in SK .

Now, what party P_i does during the round r ? It checks whether the number of signatures on this message v is at least r , including the sender's signature. If that is the case, then it accumulates that message v as a valid message in its accumulative set ACC_i , which is


initialized to empty set. And, if this message v is accumulated newly; that means, if it is the first time when this message v is accumulated during the round r . And, if there are further more rounds available in the protocol, then in the next round what this P_i does is the following. So, it already got at least r number of signed messages v , where v is identical. It has received signed v from at least r number of parties. And, it is during the round r , that P_i has accumulated this message v in its accumulative set. Then, it checks that if there are more rounds available after this round, if there are more rounds available, then in the next round what this party P_i does is the following. It relays the set of signed messages v to everyone along with its own signature on the message v .



So, that in the next round whoever not yet; whoever has not yet accumulated this message v will do that. Because, in the next round there will be now $r + 1$ number of signatures which will be available on the message v . And, if some party P_j is there who has not yet accumulated the message v , it will accumulate it; that is the idea here.

So, if I am the i th party, I have accumulated a new message and if there are rounds available to relay this information, I put my own signature on that collection of signatures. And, relay that collection of signatures to everyone as a witness that I have sufficient amount of signatures received in the previous round to accumulate the value v . So, you also do the same, that is basically the idea.



(Refer Slide Time: 34:55)

Dolev-Strong RB Protocol





$(vk_1, \dots, vk_n, sk_j)$





ACC_j



 $(vk_1, \dots, vk_n, sk_n)$

ACC_n

$(vk_1, \dots, vk_n, sk_i)$



ACC_j



 $(vk_1, \dots, vk_n, sk_1)$

ACC_1

□ Output decision for P_i :

- ❖ If ACC_i is a singleton set, then output the value in ACC_i
- ❖ Else output a default value v_0

Now, what is the decision rule for the i th party after $t + 1$ rounds? So, remember this process of collecting, accumulating messages and relaying signed messages happens for $t + 1$ number of rounds. After $t + 1$ rounds no communication will happen. The output decision for the i th party will be the following. It goes and check checks its accumulative set. If it is a singleton set, namely there is only one message which has been accumulated during the whole protocol, then that is the output value for party P_i . But, if there are multiple messages which has been accumulated by the party P_i , then it outputs some default message v_0 which could be say all 0s; that is the decision rule for the party P_i . So, that is a very simple Dolev-Strong protocol; $t + 1$ rounds of communication; first round only sender sends its signed message to everyone. As a party during the round r my step will be, I will check whether there exists a message v which has been signed at least r number of times by r different parties including the sender. If that is the case, I accumulate it and if this is the first time I am accumulating it; in the next round if it is available, I relay this information that I have accumulated v by relaying those collection of signatures along with my own signature, so that every other party who has not yet accumulated the value v , does the same in the next round. Because, it could be possible that only I have received that collection of r signatures. If sender was corrupt, it can just send a signed message along with some other parties signature on the same value to me right, but may not may send to others. But, through this relay process it is ensured that in the next round it will be accumulated by everyone else as well.

(Refer Slide Time: 37:01)

References



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

So, I have not gone through the analysis of the protocol. In the next lecture, we will do the rigorous analysis of the Dolev-Strong protocol. So, there are several texts where you can find a description of the Dolev-Strong protocol. For my presentation I have taken the description available in this research paper.

Thank you.