


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

Lecture - 33
Perfectly-Secure VSS: Necessary Condition

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:25)

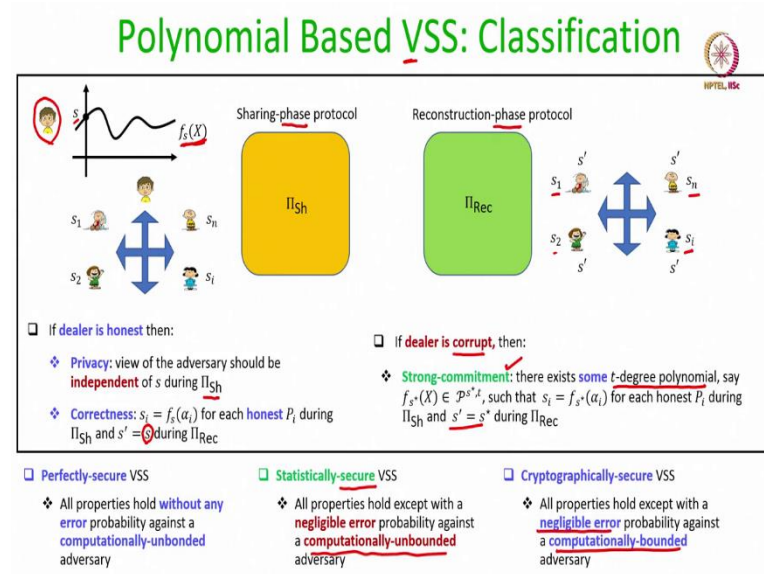
Lecture Overview



- ❑ Perfectly-secure VSS
 - ❖ Necessary condition
 - ❖ Round complexity of perfectly-secure VSS

So, in this lecture we will discuss about the necessary condition for Perfectly Secure Verifiable Secret Sharing and we will also discuss about the round complexity of perfectly secure verifiable secret sharing.

(Refer Slide Time: 00:39)



So, recall in the last lecture we had seen the definition of polynomial based VSS. Any VSS scheme will have two protocols; one for the sharing phase and another for the reconstruction phase. In the sharing phase a dealer is supposed to participate with some input s from the field and corresponding to that it will have a t -degree sharing polynomial.

And the protocol will be an interactive protocol, at the end of which every party will output a share. In the reconstruction phase every party will have its share as the input and it is an interactive protocol of at the end of which every honest party have some output s' .

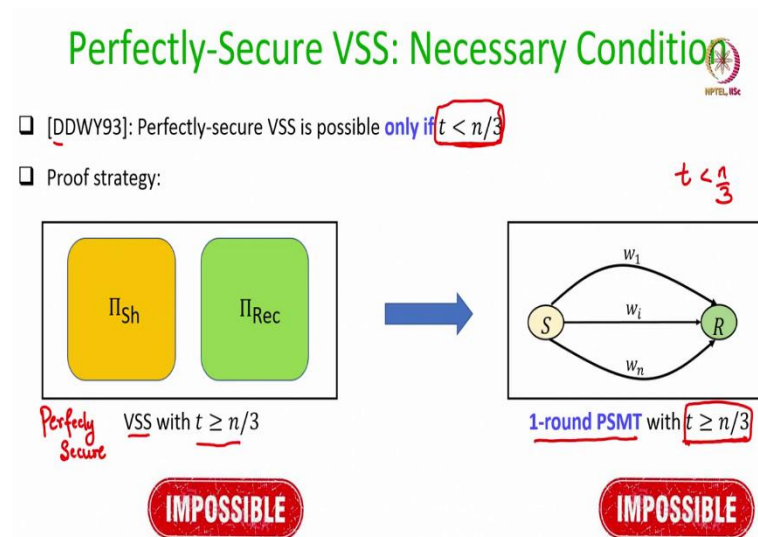
We need three properties. The privacy property which demands that if the dealer is honest and whatever adversary learns during the sharing phase that should be completely independent of the dealers input s and if the dealer is honest then the share of each party each honest party should be the evaluation of the dealer's sharing polynomial at $X = \alpha_i$. And whatever is the underlying secret which has been shared by the dealer, the same secret should be later reconstructed during the reconstruction phase.

Whereas, if the dealer is corrupt then we require the strong commitment property which demands that even if the dealer is corrupt during the sharing phase, it has shared some secret s' as per a t -degree polynomial known to the dealer and that same secret s' should later get reconstructed during the reconstruction phase.

We had seen that we have three different types of VSS schemes; the perfect perfectly secure VSS scheme where all the three security properties are achieved without any error even if the adversary is computationally unbounded. If we allow a negligible error probability in any of the three properties then the perfectly secure VSS, they get degraded to statistically secure VSS.

However, the adversary still remains computationally unbounded and in cryptographically secure VSS the adversary is computationally bounded, and a negligible error is allowed in any of the three properties.

(Refer Slide Time: 03:21)



So, now for the next few lectures our focus will be on perfectly secure VSS. So, the first question is: when exactly is it possible to design a perfectly secure VSS? We have n parties out of which t could be corrupt. Can we design a perfectly secure VSS for any value of n and t ? And answer is no. So, in a seminal work Dolev et al., have shown that perfectly secure VSS requires $t < \frac{n}{3}$, that is a necessary condition. And what is the proof strategy? Well, we will prove it by a contradiction.

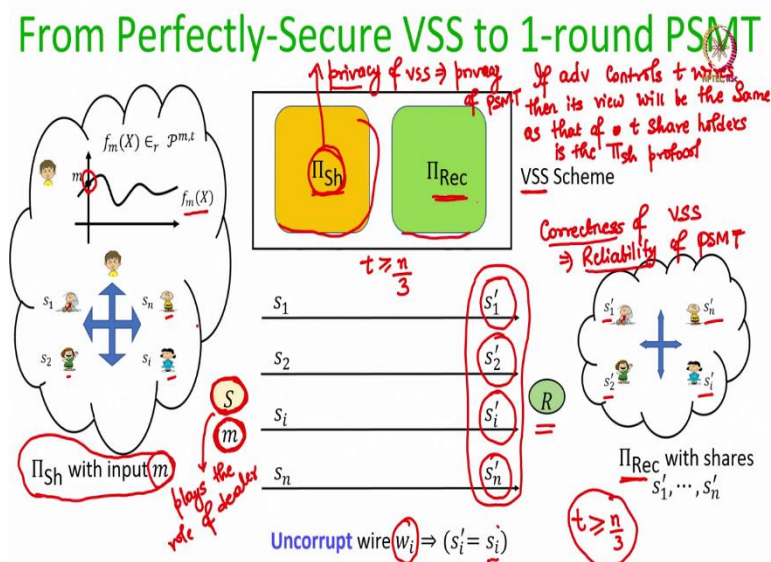
We will assume that suppose you have a perfectly secure VSS, where $t \geq \frac{n}{3}$. So, assume you have some perfectly secure VSS and when I say VSS it could be any VSS, it need not be just polynomial based. So, this condition $t < \frac{n}{3}$ it is a necessary condition not only for a polynomial based VSS, but for any VSS.

Because if it is a polynomial based VSS then you may say that ok since for reconstruction we have to apply error correction, Reed Solomon error correction and Reed Solomon error correction will work only if $t < \frac{n}{3}$ that is mean it is a necessary condition, fine. But what if my VSS scheme is not based on polynomials? It is something else. It is based on something. Else even for such VSS the condition $t < \frac{n}{3}$ is required.

So, the way we prove it is as follows. Imagine you have some perfectly secure VSS, where the condition $t < \frac{n}{3}$ is not satisfied. Say $t \geq \frac{n}{3}$. Then using this VSS scheme, we will design a one round PSMT protocol; Perfectly Secure Message Transmission protocol, where there are n wires from S to R unidirectional wires from S to R and even if up to t wires are corrupt the properties of PSMT will be achieved, where $t \geq \frac{n}{3}$.

However, recall that we know that the condition $t < \frac{n}{3}$ is necessary for one round PSMT protocol; that means, we can never design a one round PSMT protocol with this condition, where these many wires out of the n wires could get corrupt. That means, whatever we assumed regarding the perfectly secure VSS is also incorrect, that is a proof strategy.

(Refer Slide Time: 06:08)



So, everything boils down now to how we reduce the VSS problem to the PSMT problem ok; that means, how we can design a PSMT protocol assuming we have a VSS scheme. So, this is the way we can design a PSMT protocol assuming we have a VSS scheme.

So, in the PSMT protocol the sender will have some message. Now, assume that the sender's message is from some space, it could be from the secret space of the underlying secret sharing scheme. So, to send its message m in a secure way to the receiver what the sender can do is the following. It will take the protocol code of sharing phase protocol of the VSS scheme.

Now, the sender will play the role of dealer. So, it plays the role of dealer assuming that m is the secret which it would like to share among n shareholders. So, there are no shareholders. So, he is a man. The sender is playing mentally in his mind that, assume that I am the dealer with m being the secret which I want to share, and these are the n shareholders. So, it is simulating the role of those n shareholders in its mind.

So, since it is playing the role of the sender ok, so, it will think that ok it is the dealer with some secret m and to share it, it will pick some t degree polynomial. So, I am right now assuming that the VSS scheme is a polynomial based VSS scheme, but that need not be the case, this is just for simplicity.

So, it is playing the role of the dealer assuming that the dealer's input is the message m and then it plays the role of the n shareholders as per the protocol Π_{Sh} ; that means, whatever the messages those n shareholders would have exchanged among themselves during the execution of a protocol Π_{Sh} where the sender or the dealer's input would have been m that role is played by the sender here.

And once the interaction the simulated interaction among the virtual shareholders is over right, the sender will have the output of each shareholder from the protocol Π_{Sh} . So, that vector of outputs I am denoting by $s_1, s_2, \dots, s_i, \dots, s_n$. So, basically what has the sender done?

It has basically simulated it has run an instance of Π_{Sh} playing the role of the n shareholders, assuming they are virtual, and sender played the role of the dealer with input m and it basically computed the n shares which it would have distributed during the protocol Π_{Sh} .

Now, those shares it communicate to the receiver in the PSMT protocol. And how those shares are communicated? The 1st share is communicated over the first wire, the 2nd share is communicated over the second wire, the i th share is communicated over the i th wire

and the n th share is communicated over the n th wire. Now, let us see what happens if adversary controls t out of this n wires.

If adversary controls any t out of this n wires in the PSMT scheme, what exactly the adversary will learn? If adversary controls t wires, then its view will be the same as that of n is that of t shareholders in the Π_{Sh} protocol. So, for instance if adversary controls the first t wires that adversary basically learns s_1, \dots, s_t right. Now is s_1, \dots, s_t is sufficient for the adversary to learn about the message m ? And the answer is no.

Because the Π_{Sh} protocol should satisfy the privacy property of the VSS. And what is the privacy property of the VSS? It demands that even if there are up to t shareholders who get corrupt, they learn absolutely nothing about the dealer's secret in the sharing phase protocol if the dealer is honest. Now is the dealer honest? Yes, because in the PSMT scheme the sender is playing the role of the dealer and the sender is honest in the PSMT scheme or the PSMT protocol.

So, basically sender has executed an instance of the VSS scheme with an assuming an honest dealer with input m and produced n shares. Out of those n shares, t shares would have been learnt by the adversary sitting over the t channels in the PSMT protocol and those shares will be simply independent of the sender's message because sender's message is nothing but dealer's secret in the secret sharing protocol.

So; that means, the privacy property of the PSMT is preserved if sender sends the shares over the channels in the PSMT protocol. But how the receiver is going to recover the message? Because when the shares are getting communicated in the PSMT protocol, up to t wires may change those shares. They may deliver incorrect shares and receiver will have absolutely no idea which wires have delivered the correct shares and which wires have delivered incorrect shares.

So, what we can say here is that, if the wire w_i is uncorrupt it is not under adversaries control then whatever share has been delivered to the receiver is same as whatever has been sent by the sender, but as I said receiver will have absolutely no knowledge regarding whether the wire w_i is corrupt or not. So, how the receiver is going to recover? Well, we have still the reconstruction phase protocol of the VSS. So, what receiver will do is the following.

It will run an instance of Π_{Rec} in its mind that namely it will run the steps of the Π_{Rec} protocol for P_1 , it will run the steps of the Π_{Rec} protocol for P_2 it will run the steps of the Π_{Rec} protocol for P_i and it will run the steps of the Π_{Rec} protocol for P_n assuming that in the Π_{Rec} protocol P_1 has made public the share s'_1 , P_2 has made public the share s'_2 , P_i has made the share s'_i and P_n is participating with the share s'_n .

So, now what basically receiver is doing is receiver is treating each wire as a party as a virtual party as per the reconstruction protocol of the secret sharing scheme and whatever share has been received to by the receiver over that wire receiver is assuming that corresponding to that wire whatever party has been assigned the role that party is participating in an instance of the Rec protocol with that share.

So, how many parties in this instance of Rec protocol are corrupt up to t and $n - t$ are honest? Now, what is the property of the Π_{Rec} protocol? The Π_{Rec} protocol should guarantee that if the sender if the dealer during the sharing phase was honest then whatever secret has been shared during the sharing phase protocol the same secret should get reconstructed even if up to t parties are corrupt during the reconstruction phase.

Now, is the dealer honest during the execution of the sharing phase? Yes, because who played the role of the dealer? It is the sender of the PSMT protocol who played the role of the dealer. And he played the role of the dealer honestly by indeed picking a random t degree polynomial with its message being the constant term.

That means, even if up to t shares among this vector of received n shares are corrupt, once the receiver runs the Π_{Rec} protocol it should be the sender's secret which should get reconstructed namely, its message m should get reconstructed. So, what we can say is that the correctness property correctness of VSS scheme implies the reliability property of PSMT.

So, recall that for PSMT we need two security properties, privacy, and reliability. Privacy means that even if up to t channels are corrupt the adversaries view over those t channels should be independent of the senders message that we have already argued namely, the privacy of VSS implies privacy of PSMT. And now we have argued that the correctness property of VSS implies the reliability property of PSMT.

That means, if this pair of protocols Π_{Sh} , Π_{Rec} exist with $t \geq \frac{n}{3}$, then this PSMT protocol where sender plays the role of the dealer and distribute the shares and then receiver playing the role of receiver simulating n share holders and playing their role in the reconstruction protocol also constitutes a PSMT protocol a one round PSMT protocol with $t \geq \frac{n}{3}$.

But we know that there exist no one round PSMT protocol with $t \geq \frac{n}{3}$; that means, whatever we assumed regarding the existence of Π_{Sh} , Π_{Rec} is also not correct.

(Refer Slide Time: 17:52)

Some Conventions in VSS Protocols

□ System model :

- ❖ n parties, connected by pair-wise secure channels
- ❖ Internal-dealer model (Dealer is assumed to be one of the n parties)
- ❖ System-wide broadcast channel (oracle)
 - Allows a sender party to send any message identically to all (even if the sender party is corrupt)
 - Simplified abstraction
 - Broadcast oracle can be emulated by running a broadcast protocol over pair-wise secure channels for any $t < \frac{n}{3}$

So, that shows that the condition $t < \frac{n}{3}$ is necessary for any perfectly secure VSS scheme. Of course, during the proof I have used explicitly a polynomial based VSS scheme for showing the reduction or showing the equivalence between the PSMT and VSS, but as I said earlier the proof box even if we have any arbitrary perfectly secure VSS scheme.

Now, looking ahead when we will be designing the perfectly secure VSS protocols or any VSS protocol, we will follow some conventions. So, let us go through those conventions. So, what will be the system model? We will be assuming that we have n parties connected by pair-wise secure channels we will follow the internal dealer model what does that mean? That means dealer is assumed to be dealer is assumed to be one of the n parties and it will be known who is playing the role of the dealer.

There is something called external dealer model, where dealer could be some external party outside these n parties and when I say that t parties could get corrupt then in the internal dealer model it could be any t parties including the dealer, but in the external dealer model it will be t parties outside t parties excluding the dealer plus potentially including the dealer right. So, whatever protocol we designed for internal dealer model those protocol will work even for the external dealer model.

So, that is just a technicality. So, we do not have to worry about that. So, we will be sticking only to the internal dealer model. Now, the interesting thing is that apart from the pair wise secure channel we will also assume that there is a system-wide broadcast channel available to all the parties. So, what does that broadcast channel allows? So, that broadcast channel allows any sender party to send some message in the protocol identically to all the parties even if that sender party is corrupt.

Now, you might be wondering that how can such a broadcast channel be available system wide. Well, this is just a simplified abstraction while presenting the protocols or while designing the protocols because such a broadcast channel can be emulated by running a reliable broadcast protocol where all the messages are exchanged only over the pair-wise secure channels provided we are working with the condition $t < \frac{n}{3}$.

And this condition $t < \frac{n}{3}$ will be maintained whenever we will be designing any perfectly secure VSS protocol. So, this is just a simplifying assumption; that means, if we have a step in the VSS protocol where we have a step say party P_1 sends say broadcasts message x to all right. So, if I write a step like this then here, basically, I am assuming that whatever broadcast channel is available system wide, P_1 will put the message x over the channel and it will be delivered identically to all the recipients.

But when we want to implement the VSS protocol, then while implementing the VSS protocol we must implement this step. And how will we implement this step? Well, there is no broadcast channel available in the system; everything is pair wise secure channel. To implement this step what we will do is we will run any perfectly secure broadcast reliable broadcast protocol say the king-phase protocol where P_1 will be the sender with input x .

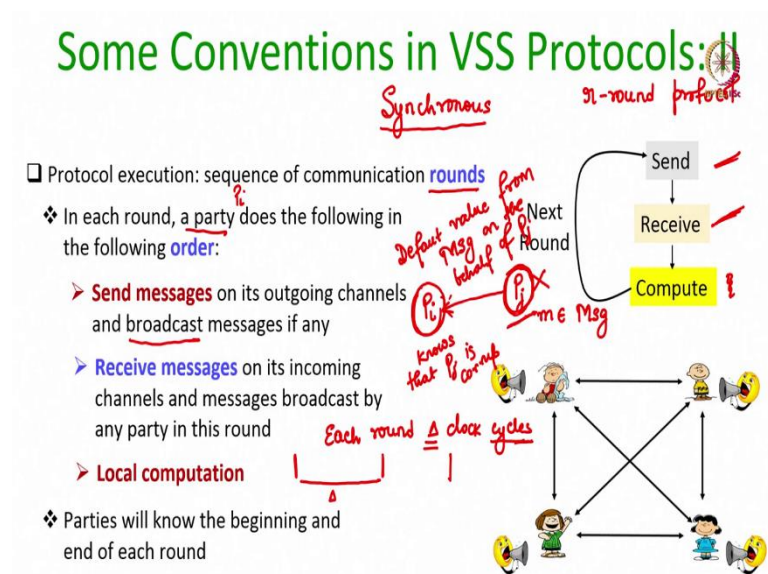
So, if P_1 is honest then it will be playing the role of the sender honestly in this phase king protocol with input x . So, at the end of the phase king protocol, the output of all the parties

will be x . Of course, if the sender is corrupt; that means, if P_1 is corrupt then the output of the phase king instance could be any message, but it will be common for all the parties that is equivalent to saying that in the VSS protocol in this step whatever P_1 has broadcasted that is a junk value.

So, this is just a simplifying assumption namely the presence of a system-wide broadcast channel. When we design statistically secure VSS and cryptographically secure VSS schemes there also we will assume that a system-wide broadcast channel is available.

And there in those protocols to emulate the broadcast channel broadcast oracle, we will have to run a statistically secure reliable broadcast protocol or a cryptographically secure reliable broadcast protocol depending upon whether the underlying VSS scheme is statistically secure or cryptographically secure. So, that is the first convention.

(Refer Slide Time: 24:01)



The second convention is regarding the protocol execution. So, we will be mostly designing VSS schemes in the synchronous communication model, where the protocol will be executed as a sequence of communication rounds. So, what will happen in each round? In each round in the protocol a party may do the following steps in the following order.

It may send some messages on its outgoing channels as dictated by the protocol depending upon its own input and whatever random coins it has during the protocol execution, and it

may also decide to some it may also decide to broadcast some message over the broadcast channel. Again, it depends upon the protocol steps.

Now, in the same round whatever messages the other parties have sent to that party over the pair wise channels and whatever other parties have broadcasted in the same round those messages will be received by every party. So, whatever I have said here is for every party P_i .

So, party P_i 's actions will be to send messages if at all it has to send any message over the pair-wise channels and if it has to broadcast something, then it broadcasts that using the broadcast channel which is assumed to be present and that is then followed by performing some local computation.

That means, whatever you have sent whatever you have received in that round based on that you compute the next set of messages as per the protocol steps and then you go to the next round. So, if we say the protocol is a r -round protocol, then the sequence of send, receive and compute will be performed r times.

That means, the parties will begin the first round they will send whatever messages they want to send to whichever party as per the protocol, in the same round whatever messages a party is supposed to receive from the other parties over the point-to-point channels and the broadcast channel it will receive and based on those messages it will decide what to compute and send in the next round. And then that will mark the beginning of the next round.

And since we are in the synchronous communication model parties will know the beginning and end of each round. So, for instance if each round is of Δ clock cycles, then the value of Δ will be publicly known and the time lapse between two consecutive rounds will be Δ clock cycles. This clock cycles could be seconds could be minutes could be hours could be days could be weeks depending upon the latency of your underlying communication network.

That means, since the parties knows the beginning and end of each round and if in the protocol P_i is expected to receive a message from P_j say either over the broadcast channel or over the point to point channel in some round k and say P_j is corrupt it decides not to send the message and the round is over then P_i will definitely know that P_j is corrupt.

So, what it can do is on the behalf of P_j it can substitute some default message. So, for instance if P_j was supposed to send a message m from some message space in that round and he has not sent the message because he was corrupt. Then what P_i can do is, it can substitute a default value from this message space Msg on the behalf of P_j and proceed to the next round.

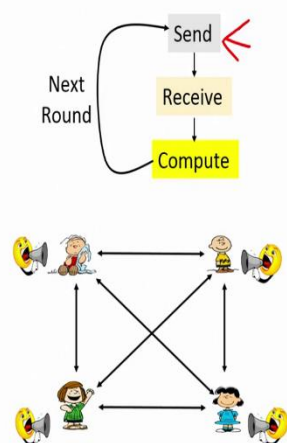
That means, we will not add extra steps where we will say if P_j has not sent and do this, do this. Whenever something is supposed to be received by a party within a round and that value does not arrives from that some specified space message space then you assume that then that definitely means the text sender party was corrupt. So, you assume that that sender party wanted to send some default message to you in that round and go to the next round.

That is equivalent to saying that if P_j was present in the system and if we would have sent the default value to P_i , then the protocol should have still achieved whatever required properties it should achieve. So, think like that and proceed to the next round. Do not do some extra computations extra calculations to handle this case.

(Refer Slide Time: 29:47)

Some Conventions in VSS Protocols: III

- ❑ Round complexity of a VSS scheme:
 - ❖ Number of communication rounds in the sharing-phase protocol
 - ❖ Reconstruction-phase usually takes a fixed number of rounds
- ❑ If dealer is (publicly) identified to be cheating during sharing-phase, then D is discarded --- parties halt the protocol and accept some default value as the secret on the behalf of dealer



The diagram illustrates a VSS round and a network of four parties. The top part shows a flowchart for a round: 'Send' (with a red arrow pointing away) leads to 'Receive' (in a yellow box), which leads to 'Compute' (in a yellow box). A curved arrow labeled 'Next Round' loops back from 'Compute' to 'Send'. The bottom part shows a network of four parties, represented by cartoon characters with megaphones, connected in a fully meshed topology with bidirectional arrows between every pair of parties.

Now what is the third convention? The third convention is regarding the round complexity of a VSS scheme since each round consists of a sequence of send, receive and compute

right. So, the round complexity of a VSS scheme is defined to be the number of communication rounds during the sharing phase of the protocol.

Namely, how many times the parties have to send messages to other parties or followed by receive and compute. This is because typically in most of the VSS schemes the reconstruction phase usually requires a fixed number of rounds. So, for instance for most of the perfectly secure VSS schemes, the reconstruction phase requires only a single round where every party just makes public its share and then parties do some error correction and reconstruct the secret shared value.

But for statistically secure VSS that need not be the case, but irrespective of the category of the VSS scheme the number of rounds required during the reconstruction phase to reconstruct underlying secret share value will be always fixed, it will not change. So, that is why we do not count the number of rounds required in the reconstruction phase, while measuring the number of rounds of the VSS scheme or while computing the round complexity of a VSS scheme.

Now, you might be wondering that why this round complexity is a complexity measure. Well, remember that every time the messages have to be exchanged the parties have to send messages over the underlying communication model. If your underlying communication model if the underlying network is the internet and if the participants are over are lying in the different parts of the world, then we will prefer a protocol where the number of times the parties have to exchange messages is less.

Namely, the number of rounds in the protocol is less right. So, that is why the round complexity is a very important measure of any distributed computing task including the VSS. Finally, one more convention that we will follow while presenting the VSS schemes.

So, whenever we will be designing a VSS scheme there will be a there will be a sequence of verification steps to verify whether dealer is following the protocol code properly or not. And if in one of those steps the parties publicly identified that dealer is cheating, then we will stop the protocol there itself, we will not proceed and we will simply say dealer is discarded.

What does “dealer discarded” mean? Dealer discarded means that the parties halt the protocol they are in itself they do not execute the rest of the protocol steps and they accept

some default value from the secret space on the behalf of the dealer as if dealer wanted to share that secret.

Of course, if the dealer is honest then an honest dealer should never be discarded during the protocol steps during the sharing phase. So, if at all the dealer is discarded during the execution of the VSS scheme, it has to be only when dealer is corrupt potentially corrupt. And if the dealer if a potentially corrupt dealer is discarded, we will end the sharing phase assuming that the party assuming that the dealer wanted to share some default value. The default value could be say 0 if the underlying sharing space is the field.

So, the default value could be 0 or it could be some publicly accepted publicly agreed upon value; that means, it will be known to everyone that, if we are discarding the dealer, then this is the public value, this is the value from the secret space underlying secret space which we will take as the dealers secret and according to that we will set our shares like that.

(Refer Slide Time: 33:59)

Interplay of Round Complexity and Fault Tolerance for Perfect VSS

$t < \frac{n}{3}$ is a
necessary condition
for any
perfectly secure

Characterization	Round Complexity
$n \geq 3t + 1, t \geq 1$	<u>3</u>
$n \geq 4t + 1, t \geq 1$	$t < \frac{n}{4}$ <u>2</u>
<u>$t = 1$</u> $n \geq 5$	<u>1</u>

of rounds in the
Sharing phase

$t \leq 4$ $t \geq 2$ ✗

❑ Resilience **decreases** as the number of **rounds decreases**

So, these are some of the conventions which we follow while designing the while writing the protocol code for VSS schemes. Now, the last thing that I want to discuss in this lecture is regarding the interplay of round complexity and fault tolerance for perfect VSS. So, recall that we have defined around complexity to be the number of rounds in the sharing phase. Also remember that $t < \frac{n}{3}$ is a necessary condition for any perfectly secure VSS.

It turns out that if we keep on decreasing the number of rounds in the sharing phase protocol; that means, if I put a restriction that ok, you are not allowed to have more than 2 communication rounds in the sharing phase protocol, then the condition $t < \frac{n}{3}$ itself is not necessary.

We require a stricter necessary condition namely the condition $t < \frac{n}{4}$. Whereas, if I allow you 3 or more number of rounds in the sharing phase then you can design a perfectly secure VSS even if up to $t < \frac{n}{3}$ corruptions.

Whereas, if I just allow you one round in the sharing phase; a one round sharing phase means only dealer computes the share and distribute the share that is all after that no interaction allowed among the parties to verify whether dealer has performed its steps properly or not.

If that kind of VSS scheme you are looking for then it is possible only if you are willing to tolerate one corruption that is all and for that there should be at least five parties present in the system, but as long as soon as I make t equal to greater than equal to 2, then you cannot design a one round secret sharing protocol, no, you cannot do that.


So, you can see there is a very nice interplay between the round complexity and the fault tolerance as you keep on decreasing the number of rounds in the sharing phase protocol; that means, you make my life more and more difficult by putting more restrictions on the number of times the parties are allowed to interact in the protocol you basically also make the resilience of the protocol worse.

With 3 or more number of rounds you can tolerate up to 33 percent corruption, with only 2 rounds being allowed at the sharing phase you can tolerate only up to 25 percent corruption and with only 1 round in the sharing phase being allowed you can design a protocol only if there is one corruption in the system out of at least 5 parties ok.

So, we will not be going through the proof for this characterization that why for 3 rounds. Of course, $n \geq 3t + 1$ is required for any round VSS protocol irrespective of whether it is 3 round, 2 round, 1 round. But the proof that, but for the, but the proof for this condition that for a 2 round perfectly secure VSS we require $t < \frac{n}{4}$ the proof is slightly involved, it requires a lot of new things to be discussed which due to interest of time.

(Refer Slide Time: 38:05)

References



- ☐ D. Dolev, C. Dwork, O. Waarts and M. Yung: Perfectly Secure Message Transmission. JACM 40(1): 17-47, 1993
- ☒ Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, Tal Rabin: The round complexity of verifiable secret sharing and secure multicast. STOC 2001: 580-589
- ☐ Anirudh Chandramouli, Ashish Choudhury, Arpita Patra: A Survey on Perfectly-Secure Verifiable Secret-Sharing. ACM Computing Surveys 2022

I will be not going into it, but if you are interested to know more about the proof for the round complexity of perfectly secure VSS you are referred to this nice paper. And proof for the necessity condition of $t < \frac{n}{3}$ for perfectly secure VSS is given in this seminal work by Dolev et al., where they introduced the problem of PSMT and as I said earlier that I have a survey paper on perfectly secure VSS schemes.

If you want to know a lot more about perfectly secure VSS schemes, you are referred to this survey.

Thank you.