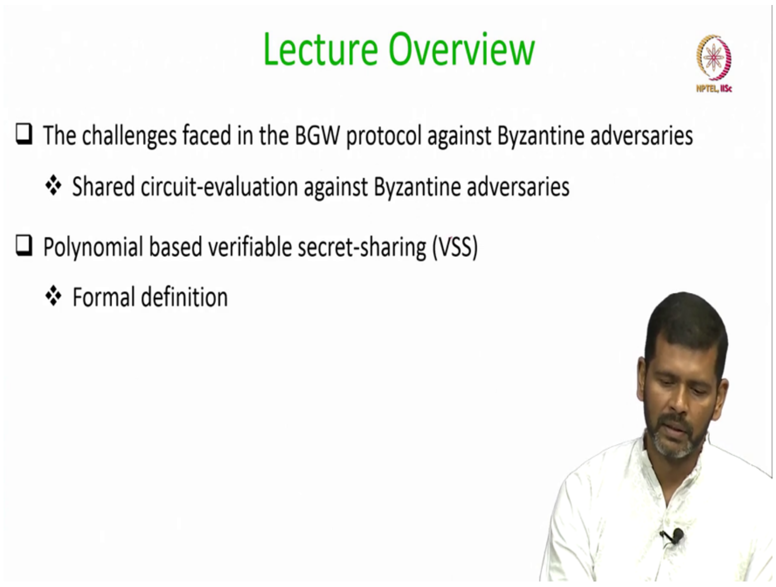
Secure Computation: Part II

Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Information Technology, Bangalore

Lecture - 32
The BGW MPC Protocol for Byzantine Corruption : Challenges

Hello everyone, welcome to this lecture. So, in this lecture we will see what are the challenges we face when we run the BGW MPC Protocol against Byzantine Corruptions ok.

(Refer Slide Time: 00:32)

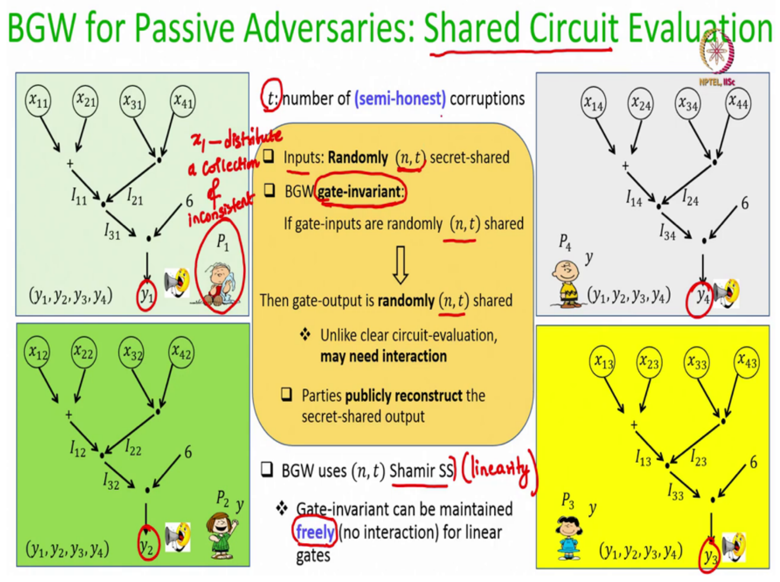


Lecture Overview

- ❑ The challenges faced in the BGW protocol against Byzantine adversaries
  - ❖ Shared circuit-evaluation against Byzantine adversaries
- ❑ Polynomial based verifiable secret-sharing (VSS)
  - ❖ Formal definition

So, after we introduce the challenges that we face while running the BGW protocol against Byzantine adversaries, we will go through the definition of verifiable secret sharing which will be one of the main ingredients which will be used later when we study the BGW protocol against Byzantine adversaries.

(Refer Slide Time: 01:02)



BGW for Passive Adversaries: Shared Circuit Evaluation

t: number of (semi-honest) corruptions

- Inputs: **Randomly** (n, t) secret-shared
- BGW gate-invariant:

  If gate-inputs are randomly (n, t) shared

  $\Downarrow$

  Then gate-output is **randomly** (n, t) shared

  - Unlike clear circuit-evaluation, **may need interaction**
- Parties **publicly reconstruct** the secret-shared output

- BGW uses (n, t) Shamir SS (linearity)
  - Gate-invariant can be maintained freely (no interaction) for linear gates

x_1 — distribute a collection of inconsistent

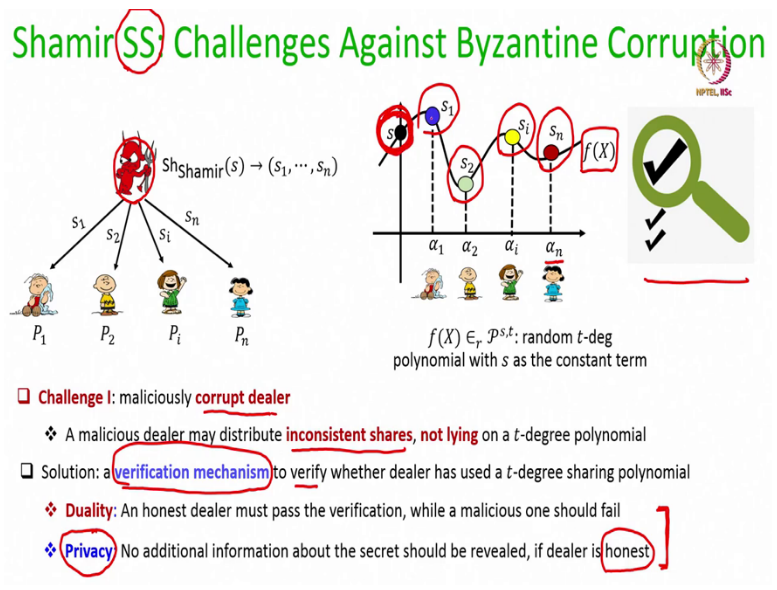
x_{11} x_{21} x_{31} x_{41} I_{11} I_{21} I_{31} 6 P_1 (y_1, y_2, y_3, y_4) y_1
x_{12} x_{22} x_{32} x_{42} I_{12} I_{22} I_{32} 6 P_2 y (y_1, y_2, y_3, y_4) y_2
x_{13} x_{23} x_{33} x_{43} I_{13} I_{23} I_{33} 6 P_3 y (y_1, y_2, y_3, y_4) y_3
x_{14} x_{24} x_{34} x_{44} I_{14} I_{24} I_{34} 6 P_4 y (y_1, y_2, y_3, y_4) y_4
NPTEL IISc

So, recall that in the last lecture we had seen the idea behind the shared circuit evaluation followed in the BGW protocol against passive corruptions, where up to t-parties can be under the control of a semi honest adversary ok. So, the idea there was that the input providers secret share their respective inputs and then each gate is evaluated in a secret shared fashion where if the gate inputs are secret shared in a t out of n secret shared fashion then the gate output is also made available to the parties in a t out of n secret shared fashion.

And once the parties have shares for the function output they make those shares public and publicly reconstruct the function output. We also discussed in the last lecture that BGW protocol uses t out of n secret sharing. This is because the t out of n secret sharing has this linearity property which we also had seen in the last lecture and due to that linearity property maintaining the BGW gate invariant is completely non interactive for linear gates, ok.

Namely to evaluate the linear, the parties have to perform no interaction at all ok.

(Refer Slide Time: 02:45)



Shamir SS: Challenges Against Byzantine Corruption

$f(X) \in_r \mathcal{P}^{s,t}$: random t-deg polynomial with s as the constant term

- Challenge I: maliciously corrupt dealer
  - A malicious dealer may distribute inconsistent shares, not lying on a t-degree polynomial
- Solution: a verification mechanism to verify whether dealer has used a t-degree sharing polynomial
  - Duality: An honest dealer must pass the verification, while a malicious one should fail
  - Privacy: No additional information about the secret should be revealed, if dealer is honest

Now, let us see what are the challenges we face when we run the BGW protocol against Byzantine corruptions. Namely, where the adversary or the t corrupt parties may not follow the protocol instructions and the main challenge is basically to ensure that the properties of Shamir Secret Sharing are maintained that is one of the major challenges of course, some other challenges are also there which we will discuss later on.

Recall that the BGW protocol uses t out of n Shamir secret sharing, but t out of n Shamir Secret Sharing scheme will work only in the passive or the semi honest corruption model. What we are now going to discuss are the challenges which we face while designing or while running the Shamir Secret Sharing in the Byzantine corruption model. So, what is the Shamir Secret Sharing scheme or Shamir's secret sharing mechanism?

If there is a secret s which needs to be secret shared by the dealer then the dealer has to pick a random t-degree polynomial whose constant term is the secret to be shared and each party is supposed to be provided the value of that polynomial at a fixed point alpha i ok. So, the ith party is supposed to be provided the value of that polynomial at alpha i, nth party is supposed to be provided the value of that polynomial at alpha n and so on.

If the dealer is honest namely the code or the if the system or the computer of the dealer is not corrupted, then indeed the dealer will pick a t-degree polynomial and the constant term of that polynomial will be the secret of the dealer and that polynomial will be evaluated at alpha 1, alpha 2, alpha i and alpha n and the ith point will be given to the ith party.

Now, let us see what goes wrong, if the same protocol is executed in the Byzantine corruption model where now the adversary may corrupt up to t-parties potentially including the dealer and where the corrupt parties may not follow the protocol instructions. So, now, if the dealer gets corrupt during the execution of the Shamir Secret Sharing then what a corrupt dealer may do is that it may not pick a t-degree polynomial at the first place and it may simply distribute inconsistent shares to the parties.

Namely the share s1, the share s 2, the share si and the shear sn may not collectively lie on a t-degree polynomial ok. So, that is one of the challenges that we face when we run the Shamir Secret Sharing against the Byzantine corruption model and note that the parties will not be knowing whether the dealer is following the protocol instructions or not if we are running this protocol in the Byzantine corruption model because we do not know who are the corrupt parties in the system.

We only have an upper bound namely t on the maximum number of corruptions which can happen. That means, if we are now trying to run this Shamir Secret Sharing protocol in the Byzantine corruption model, then once the dealer distributes the points on the polynomial f(X) we have to incorporate a verification mechanism, the party specifically have to incorporate a verification mechanism. We should verify whether the dealer has used a single t-degree sharing polynomial while computing and distributing the shares to the respective parties ok.

And this verification mechanism will have a duality property in the sense that an honest dealer should always pass the verification because if the dealer is not corrupt then of course, it will be following the code as per the Shamir Secret Sharing protocol. So, whatever verification mechanism we are incorporating on top of the basic Shamir Secret Sharing scheme that verification mechanism should pass if the dealer is not under adversary's control.

Whereas, if the dealer is maliciously corrupted and is not following the code of Shamir secret sharing, then that verification should be able to catch such a corrupt dealer. Now you might be wondering what is the big deal in verifying whether dealer has used a t-degree polynomial for computing the shares, the parties can simply make their shares public, P1 can say hey I have got s1, P2 can say I have got s2, Pi can say I have got si and Pn can say I have got sn and when I say that they are saying, I mean to say they are making their shares public.

And then everyone can check whether the shares s1, s 2, s i, sn lie on a t-degree polynomial or not fine. That could be one of the verification mechanisms, but the big problem with that verification mechanism is that it completely breaches the privacy required from the Shamir secret sharing. The privacy property required from the Shamir Secret Sharing scheme is that, if any t shares are available then adversary should not be able to compute anything about the underlying secret s, but in this verification mechanism what we are doing?

We are actually making all the shares public. So, if the dealer would have been honest right, if the dealer would have been honest then through this verification mechanism where all the parties simply make their shares public, the privacy of the secret will be lost ok. So, we cannot incorporate such a naive verification mechanism. So, whatever is the verification mechanism which we incorporate on top of the summer secret sharing scheme that verification mechanism now has two challenging goals.

It should be able to catch a bad dealer who is not following the protocol instructions and who is not distributing shares lying on a t-degree polynomial and at the same time during that verification process the privacy of the secret s should not be lost, it should be preserved if the dealer is honest and remember the parties will not be knowing whether the dealer is honest or corrupt.

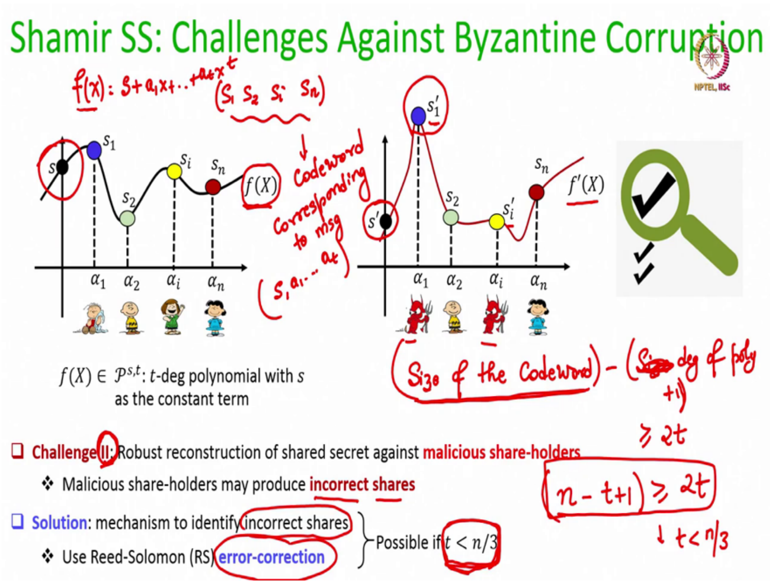
So, whatever verification mechanism we propose on top of the basic Shamir Secret Sharing that should work irrespective of whether the dealer is honest or corrupt. You might be wondering that well if the dealer gets corrupt then what advantage it gets, it will anyhow learn the dealer secret with adversary will learn the dealer secret if adversary corrupts the dealer fine.

But the harm that it will cause is that it will no longer be ensured that the shares of the parties lie on a t-degree polynomial and this will have a significant impact when we try to maintain the BGW gate invariant in the MPC protocol.

The gate invariant basically starts with the assumption that inputs of all the parties are secret shared as per a t out of n secret sharing mechanism, but now if we run the Shamir Secret Sharing scheme in the Byzantine corruption model and say P1 gets corrupt in a Byzantine fashion, then when it is distributing the shares of x1 it can simply distribute a collection of junk shares or inconsistent shares which do not lie on a t-degree polynomial

That means, this invariant will not be maintained now if we try to run the BGW protocol in the malicious model because the inputs at the first place are not secret shared as per a t out of n Shamir Secret Sharing scheme instance ok. So, that is the first challenge which we face when we try to implement when we try to execute the BGW protocol in the Byzantine corruption model.

(Refer Slide Time: 12:02)



Shamir SS: Challenges Against Byzantine Corruption

$f(X) \in \mathcal{P}^{s,t}$: t-deg polynomial with s as the constant term

- Challenge II: Robust reconstruction of shared secret against malicious share-holders
  - Malicious share-holders may produce incorrect shares
- Solution: mechanism to identify incorrect shares
  - Use Reed-Solomon (RS) error-correction

Possible if $t < n/3$

$f(x): s + a_1x + \ldots + a_tx^t$
$(s_1\ s_2\ s_i\ s_n)$
Codeword corresponding to msg
$(s, a_1, \ldots a_t)$
s
s_1
s_2
s_i
s_n
$f(X)$
α_1 α_2 α_i α_n
s'
s_1'
s_i'
$f'(X)$
NPTEL, IISc
(Size of the Codeword) - (deg of poly +1)
$\geq 2t$
$(n - t+1) \geq 2t$
$t < n/3$

The second challenge again is associated with Shamir Secret Sharing scheme when we run it against the Byzantine corruption model. So, the challenge is during the reconstruction phase. So, assume for the moment that the dealer is honest and it has indeed followed the code of Shamir Secret Sharing scheme namely, it has picked a t-degree polynomial and distributed points or distributed shares lying on a t-degree polynomial and now suppose t+1 shareholders come together they want to reconstruct the secret ok.

Now, at this point when the secret is getting reconstructed it may so happen that adversary corrupts t shareholders in a Byzantine fashion, in a malicious fashion and those malicious shareholders now produce incorrect shares. So, for instance suppose the first shareholder and nth shareholder they are part of that group of t + 1 share-holders who are making their shares public to reconstruct the underlying secret.

But P1 gets corrupt and Pi gets corrupt and P1 produces an incorrect share s_1' and say Pi produces an incorrect share s_i'. Again, the honest parties in the system or the honest parties in that group of t + 1 parties, they will have absolutely no idea who are the corrupt shareholders in that group of t + 1 parties, they will not be knowing whether P1 and Pi are corrupt or not.

For them it will look like as if the share of P1 is $s_1^{'}$ and the share of Pi is $s_i^{'}$ and now if we use those t + 1 shares where some of the shares are incorrect shares then the interpolated t-degree polynomial will be a completely wrong polynomial and when we take the constant term of that polynomial as the output that output is not the intended output.

That means, now we need another verification mechanism here to identify the incorrect shares at the time of the reconstruction. Well, it turns out that this second challenge is relatively easier to deal with because recall that in the earlier lectures we had seen that there is a very nice relationship between Shamir Secret Sharing and Reed-Solomon error correction. So, the problem that we are right now encountering namely the challenge two is to get back the original polynomial f(X). And the original constant term s, even in the presence of incorrect shares or incorrect points or malicious shares can be taken care through Reed-Solomon error correction provided we are working with the condition t < n/3. Why the condition t < n/3 is required?

Because remember we require this condition to hold to error correct up to t errors on a t-degree polynomial namely, the size of the codeword minus the size or minus the degree of the polynomial plus 1 should be greater than equal to 2 times the number of errors that we want to error correct. So, if we consider a modified reconstruction protocol where all the n parties make their shares public; that means, the size of the codeword originally was n because there were n shareholders.

So, $(s_1, \cdots, s_n)$ is the codeword, Reed-Solomon code word corresponding to the message (s, a1, …, at). Now what are a1, …, at? Well they are the random coefficients of f(x) polynomial ok. So, if all the n parties make their shares public, it is equivalent to saying that each party is now making its corresponding component of the Reed Solomon code word public and what was the size of the polynomial? Size of the polynomial is how many coefficients were there, t + 1.

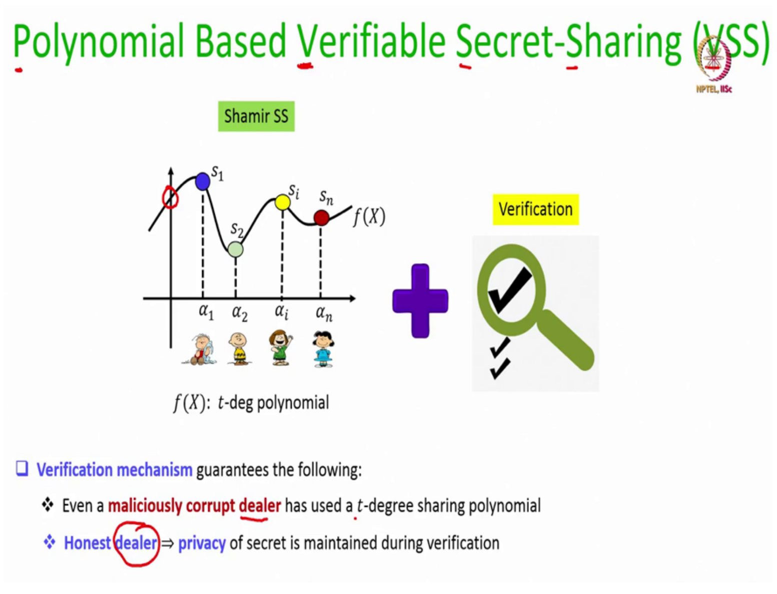
So, this should be greater than equal to 2 times the number of errors which we want to error correct. So, what are errors we want to error correct? Namely the corrupt shares which could be provided by the corrupt shareholders. So, if this condition is ensured then only the

Reed-Solomon error correction will be able to error correct the t incorrect shares provided by the corrupt shareholders and we can hope to reconstruct back the original polynomial and hence the constant term.

So, this condition automatically implies that t is less than n over 3 is required ok. So, this is the second challenge when we run the Shamir Secret Sharing in the Byzantine corruption model ok, but it turns out that if it is guaranteed that the underlying secret is secret shared through a t-degree polynomial, challenge two is relatively easier to handle. Namely instead of asking any t + 1 parties to make their shares public and use those t + 1 shares to reconstruct the secret s. Now, what we will do is that, we will ask all the n parties to make their shares public ok and even if up to t shareholders provide incorrect shares, we will be able to error correct them provided this condition is maintained, this condition is satisfied. Later on we will see that this condition is indeed a necessary condition if we want to design a perfectly secure MPC protocol in the Byzantine corruption model.

So, it is not necessary, just to handle the challenge two, it is a necessary condition imposed by any general function, if we want to compute it securely in a perfectly secure fashion ok. So, now, we had seen two challenges which we encounter in the context of Shamir Secret Sharing if we execute it in the Byzantine corruption model.

(Refer Slide Time: 19:35)



Polynomial Based Verifiable Secret-Sharing (VSS)

NPTEL, IISc
Shamir SS
s_1
s_2
s_i
s_n
$f(X)$
α_1
α_2
α_i
α_n
$f(X)$: t-deg polynomial
Verification

- Verification mechanism guarantees the following:
  - Even a maliciously corrupt dealer has used a t-degree sharing polynomial
  - Honest dealer $\Rightarrow$ privacy of secret is maintained during verification

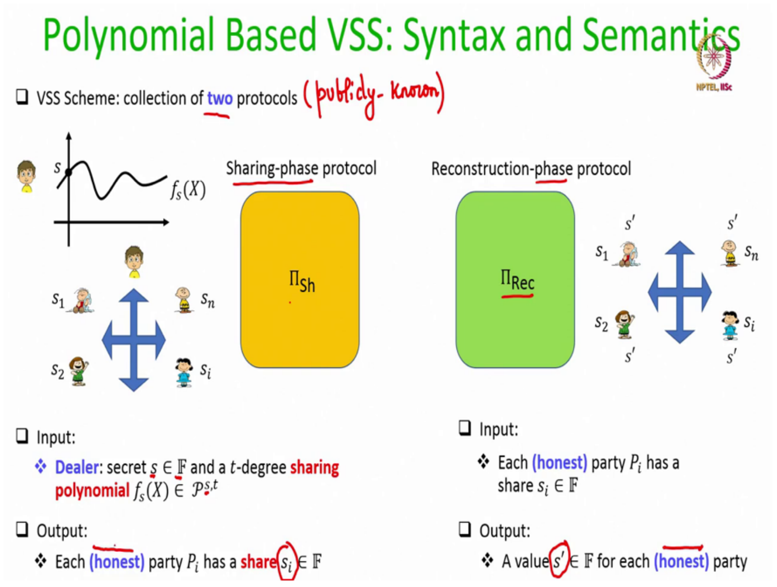
And to deal with these two challenges we have to do something more on top of the Shamir Secret Sharing and that resultant primitive is called as verifiable secret sharing ok. So, we have secret sharing S and S. And since we are now doing a secret sharing in a verifiable fashion that is why it is called VSS, Verifiable Secret Sharing, which in itself is a very fundamental primitive in secure distributed computing.

We will be studying polynomial based verifiable secret sharing because we are interested to design or modify the BGW protocol so, that it works even against Byzantine adversaries and we will stick to polynomial based secret sharing where the secrets are shared using polynomials. So, that is why we will be focusing on polynomial based VSS, but in general the VSS need not be polynomial based ok.

So, what is this polynomial based verifiable secret sharing? So, it will be your secret Shamir Secret Sharing where the dealer should pick a t-degree polynomial, a random t-degree polynomial with the secret being the constant term, followed by some verification mechanism. And that verification mechanism informally should guarantee the following properties. It should guarantee that even the dealer is maliciously corrupt during the execution of the protocol, it has used a t-degree sharing polynomial.

That means, whatever sharing polynomial it is using during the steps of the Shamir Secret Sharing that polynomial is a t-degree sharing polynomial, if it is not a t-degree sharing polynomial then this verification mechanism will be able to catch such a maliciously corrupted dealer. And that verification mechanism guarantees that if the dealer is honest; that means, if the dealer is not under the adversary’s control then the privacy of the secret is maintained during this verification process it will not be lost.

(Refer Slide Time: 21:51)



Polynomial Based VSS: Syntax and Semantics

- VSS Scheme: collection of two protocols (publicly-known)

- Input:
  - Dealer: secret $s \in \mathbb{F}$ and a t-degree sharing polynomial $f_s(X) \in \mathcal{P}^{s,t}$
- Output:
  - Each (honest) party P_i has a share $s_i \in \mathbb{F}$

- Input:
  - Each (honest) party P_i has a share $s_i \in \mathbb{F}$
- Output:
  - A value $s' \in \mathbb{F}$ for each (honest) party

So, let us now see the syntax and semantics of any polynomial based vss. So, when I say a polynomial based VSS scheme, it is basically a collection of two protocols. The two protocols will be publicly known and one of the protocols will be used for sharing the secret. So, let us give it the name Π_{sh}. So, it is a sharing phase protocol and other protocol will be used for reconstructing the secret, we give it the name Π_{REC}.

So, there will be two phases in a VSS scheme a sharing phase and a reconstruction phase. Well, depending upon your underlying application where we are using the VSS there may not be a gap between the sharing phase and the reconstruction phase. So, for instance the dealer goes and shares the secret verification happens and once it is confirmed that verification is successful the parties go and immediately run the reconstruction protocol to reconstruct the secret. We may have an application scenario where this is the way VSS is used.

But in general for the MPC where we are going to use VSS for the shared circuit evaluation the dealer will be sharing the secret, the dealers will be the respective owners of the inputs of the function to be securely computed and then the entire circuit will be evaluated in a secret shared fashion and once the function output is available in a secret shared fashion the reconstruction phase protocol will be executed ok.

So, right now we do not want to go into those details, how much gap will be there between the sharing phase and the reconstruction phase, you can imagine that they are two independent phases. In the sharing phase the secret is shared and once the secret has been shared, when the reconstruction phase is triggered the underlying shared value is going to be reconstructed and everything will happen in the presence of up to t Byzantine corruptions.

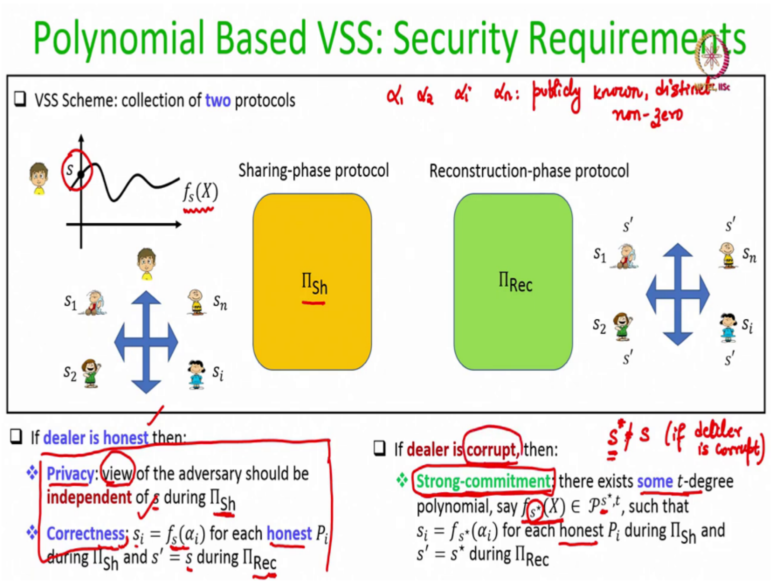
So, in the sharing phase protocol there will be a dealer, a designated dealer and it is the dealer who has the input there, no other parties will have any input. Well, they may have random inputs as part of the protocol, but they may not have any external input. So, the dealer will have a secret s from a field which is the input or the private input for the dealer and corresponding to that secret, a t-degree sharing polynomial, ok, whose constant term is the input of the dealer.

Now, this will be an interactive protocol, this protocol Π_{sh}, where dealer will distribute the shares for its secret and on top of that there will be some verification steps which will require interaction among the parties. And after the interaction is over the output for each party will be a share, ok, which is going to be a field element. So, let us call the share for the ith party let us denote the share for the ith party as si.

Now, in the reconstruction phase protocol the input for each party will be share ok and there will be an interaction among the parties and based on the interaction which happens among the parties and the steps of the protocol, each honest party in the system will output a value $s^{'}$. I stress here that when I am saying the output for the sharing phase and for the reconstruction phase I am focusing only on the output for the honest parties because we do not care what the corrupt parties output at the end of the sharing phase and the reconstruction phase.

Because since they are not going to listen to us listen to us, in the sense, listen to the protocol instructions, they may output whatever they feel like. It is only the honest parties who will stick to the protocol instructions of Π_{sh} and Π_{REC} and that is why we stick or we characterize what is required from the output of only the honest parties ok. So, that is the syntax and semantics of any polynomial based VSS.

(Refer Slide Time: 26:35)



Polynomial Based VSS: Security Requirements

- VSS Scheme: collection of two protocols

$\alpha_1\ \alpha_2\ \alpha_i\ \alpha_n$: publicly known, distinct non-zero

s
$f_s(X)$
Sharing-phase protocol
Π_{Sh}
Reconstruction-phase protocol
Π_{Rec}
s_1
s_2
s_n
s_i
s'

- If dealer is honest then:
  - Privacy: view of the adversary should be independent of s during Π_{Sh}
  - Correctness: $s_i = f_s(\alpha_i)$ for each honest P_i during Π_{Sh} and $s' = s$ during Π_{Rec}
- If dealer is corrupt, then:
  - Strong-commitment: there exists some t-degree polynomial, say $f_{s^\star}(X) \in \mathcal{P}^{s^\star,t}$, such that $s_i = f_{s^\star}(\alpha_i)$ for each honest P_i during Π_{Sh} and $s' = s^\star$ during Π_{Rec}

$s^\star \neq s$ (if dealer is corrupt)

Now, we require some security properties. So, what are the security properties we require from any polynomial based VSS? So, let us first consider the case when dealer is honest. Remember during the execution of any VSS scheme up to t-parties could get corrupt and among those t-parties dealer could be one of the corrupt parties and we do not know whether the dealer is getting corrupt or not during the run time ok. Irrespective of whether the dealer is honest or corrupt we need some properties.

So, let us first categorize or let us first enumerate the properties which we require from any VSS scheme, if the dealer is honest. If the dealer is honest then the security properties that we require are more or less what we require from Shamir Secret Sharing scheme. The first property is the privacy property namely, the adversary who can control or corrupt up to t-parties during the sharing phase should not learn anything about the secret s, namely its view should be independent of the dealers secret s and what does view of the adversary mean?

The view of the adversary mean whatever information the t corrupt parties have during the execution of the protocol, whatever messages they send, whatever messages they receive and so on. I stress that the privacy is required only for the sharing phase, not for the reconstruction phase because anyhow in the reconstruction phase the parties will be reconstructing the dealers secret and that time everyone will know what was dealer’s secret.

So, that time we cannot ask for the privacy of the dealer’s secret because everyone is supposed to learn the dealer's secret during the reconstruction phase. If the dealer is honest we also require the correctness property which demands that the share for the ith party, if the ith party is honest, should be the value of the polynomial, the sharing polynomial picked by the dealer, at x = alpha i.

So, here alpha 1 alpha 2 alpha i alpha n they are some publicly known distinct non zero values from the field ok. So, as I said these two properties are precisely the properties which we require if dealer is honest and we are running an instance of Shamir Secret Sharing. Because if the dealer is honest and if we are running an instance of Shamir Secret Sharing then indeed the share of the ith party will be the value of the dealer’s polynomial at x equal to alpha i ok.

Now in this correctness property we also have another condition that at the end of the sharing phase the share of the ith party should be the value of the dealer’s polynomial at x equal to alpha i and the value which is reconstructed during the reconstruction phase should be the dealers secret, it should not be any other value.

That means, during the reconstruction phase even if up to t-parties produce wrong shares still the value which gets reconstructed is the dealers secret, no other value should get reconstructed. So, that is the additional property compared to the Shamir Secret Sharing scheme in the semi honest model which we have in as part of the correctness property here. So, these are the two properties which we require if the dealer is honest from any polynomial based VSS scheme.

Now, let us see what property we require if the dealer is corrupt during the execution of a v polynomial based VSS scheme. That property is called as the strong commitment property and this is a very strong property ok what exactly is this property? This property demands that even if the dealer is corrupt during the execution of the VSS scheme, it should be guaranteed at the end of the sharing phase that there is some t-degree polynomial used by the dealer for sharing the constant term of that polynomial.

Namely, there exists some t-degree polynomial, let us denote that polynomial as $f_s^{\star}$ whose constant term will be $s^{\star}$, such that at the end of the sharing phase the share for each honest P i is the value of that t-degree polynomial at x equal to alpha i. And during the reconstruction phase it is the constant term of that polynomial which gets reconstructed.

Well, this property is equivalent to saying that even a corrupt dealer is following the VSS scheme following the instructions of the Π_{SH} protocol, by selecting some t-degree polynomial. Well he is supposed to pick a t-degree polynomial depending upon his input s right. But he may not decide to do that, ok, he may decide that ok I am not going to follow a t-degree polynomial, well that should not be the case.

Even if the dealer is corrupt the VSS protocol should guarantee that during the execution of the Π_{SH} protocol dealer has distributed shares lying on some t-degree polynomial of his choice such that the constant term of that polynomial, let us denote it by $s^{\star}$, where $s^{\star}$ could be different from s if the dealer is corrupt.

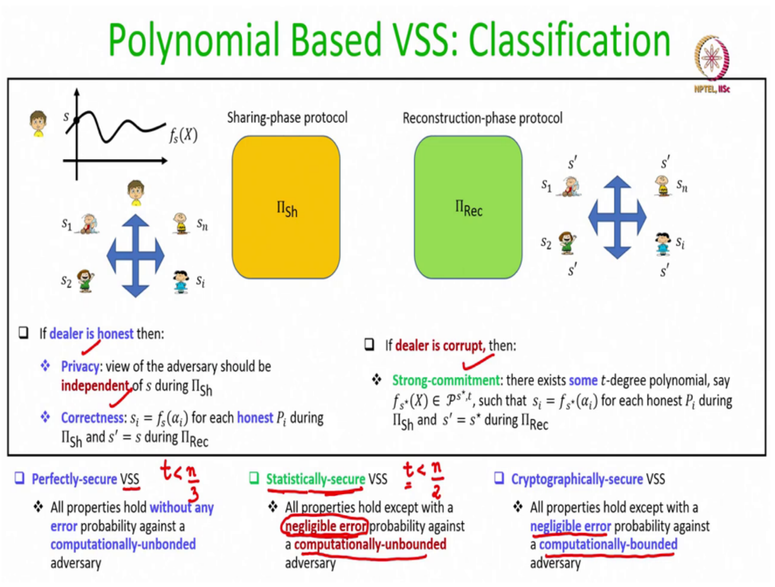
Because if dealer is corrupt then as the protocol proceed, it may decide to change its mind and say no I am not interested in sharing s, I am interested in sharing $s^{\star}$. Fine, if you change your mind, but once you change your mind to $s^{\star}$ and once you have fixed a t-degree polynomial you should stick to that and you should distribute shares to the honest parties as per that t-degree polynomial. If a corrupt dealer is not doing that then it should be caught during the execution of the sharing protocol ok.

So, that is the strong commitment property. Why it is called strong commitment? Because it enforces even a potentially corrupt dealer to commit to a t-degree polynomial. Commit to a t-degree polynomial means, picking a t-degree polynomial for sharing some secret known to the dealer and later on during the reconstruction phase, it will be the constant term of the same committed t-degree polynomial which will get reconstructed even if the dealer along with other corrupt parties try to influence the other honest parties ok.

So, it is this stronger strong commitment property which makes the VSS verifiable secret sharing scheme a very important building block because if the dealer is honest anyhow we will get the privacy and correctness namely everything will be secret shared as per a t-degree polynomial and whatever has been shared will be reconstructed.

But if the dealer is corrupt then you do not have to worry. Even in that case it will be guaranteed that whatever has been shared it has been shared as per some t-degree polynomial and the same shared thing will get later reconstructed during the reconstruction phase.

(Refer Slide Time: 34:40)



Polynomial Based VSS: Classification

NPTEL, IISc
s
$f_s(X)$
Sharing-phase protocol
Π_{Sh}
Reconstruction-phase protocol
Π_{Rec}
s_1
s_2
s_i
s_n
s'

- If dealer is honest then:
  - Privacy: view of the adversary should be independent of s during Π_{Sh}
  - Correctness: $s_i = f_s(\alpha_i)$ for each honest P_i during Π_{Sh} and $s' = s$ during Π_{Rec}
- If dealer is corrupt, then:
  - Strong-commitment: there exists some t-degree polynomial, say $f_{s^\star}(X) \in \mathcal{P}^{s^\star, t}$, such that $s_i = f_{s^\star}(\alpha_i)$ for each honest P_i during Π_{Sh} and $s' = s^\star$ during Π_{Rec}

- Perfectly-secure VSS $t < \frac{n}{3}$
  - All properties hold without any error probability against a computationally-unbonded adversary
- Statistically-secure VSS $t < \frac{n}{2}$
  - All properties hold except with a negligible error probability against a computationally-unbounded adversary
- Cryptographically-secure VSS
  - All properties hold except with a negligible error probability against a computationally-bounded adversary

So, we have discussed the security requirements for any polynomial based VSS, namely we require the privacy, correctness and the strong commitment property. Now we have three different types of VSS schemes the first category is perfectly secure VSS, where all the three properties namely privacy, correctness and strong commitment they should be achieved without any error probability even if the adversary is computationally unbounded.

The next category is statistically secure VSS where the adversary is still computationally unbounded, but now we allow a very negligible error probability in these three properties; that means, privacy property is allowed to be violated; that means, we allow the adversary to learn something about the dealer’s secret.

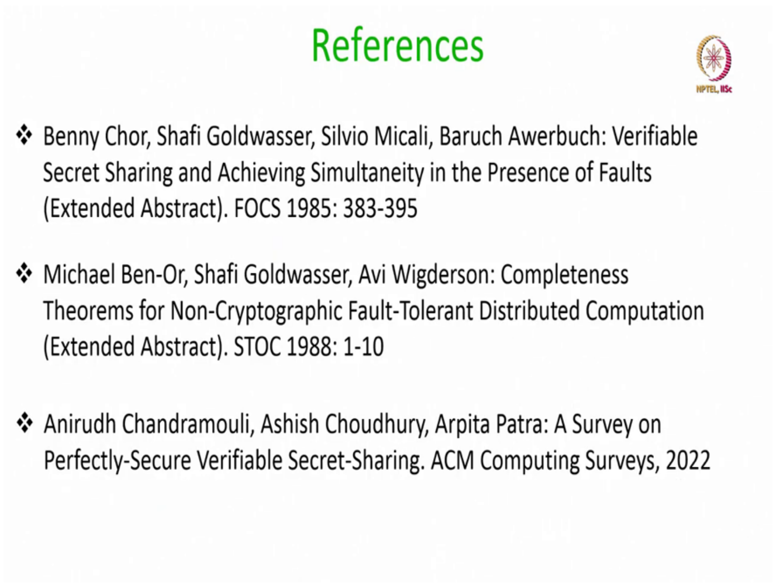
But that probability of that should be very very small it should be negligible or it could be possible that during reconstruction the honest dealer’s secret does not get reconstructed, something else get reconstructed, but again the probability of that should be negligible. Or it could be possible that the strong commitment property is violated if the dealer is corrupted, but again the probability of that should be negligible.

So, such kind of VSS such is called as a statistically secure VSS. Now you might be wondering why one will go for statistically secure VSS if we have a perfectly secure VSS? Well later we will see that there is an inherent tradeoff regarding the resilience bounds, the number of rounds required, the communication complexity for perfectly secure VSS and statistically secure VSS.

So, for instance perfectly secure VSS requires t less than n by 3 condition; that means, we can design a perfectly secure VSS only if up to 33 percent of the parties are corrupt. But we can design statistically secure VSS even if t is less than n over 2; that means, even if up to 49 percent of the system participants get corrupt; that means, we can tolerate more faults at the expense of allowing a very very small, but non zero error in the properties.

And the third category of VSS is cryptographically secure VSS where now the adversary is computationally bounded; that means, it cannot perform exponential amount of computation and where a negligible, but non zero error probability is allowed in any of these three properties ok. So, we will see all these three different types of VSS schemes as the course proceeds.

(Refer Slide Time: 37:51)



References

- Benny Chor, Shafi Goldwasser, Silvio Micali, Baruch Awerbuch: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). FOCS 1985: 383-395
- Michael Ben-Or, Shafi Goldwasser, Avi Wigderson: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). STOC 1988: 1-10
- Anirudh Chandramouli, Ashish Choudhury, Arpita Patra: A Survey on Perfectly-Secure Verifiable Secret-Sharing. ACM Computing Surveys, 2022

So, with that I end this lecture these are the references used. So, the VSS primitive as a general cryptographic primitive was introduced in this seminal work of Chor et al and the first perfectly secure polynomial based VSS was proposed in the work of BGW itself and I have a survey paper on perfectly secure VSS which has been recently published in ACM Computing Surveys.

So, that survey covers all the existing perfectly secure VSS schemes in the literature, ok, both in the synchronous as well as in the asynchronous communication setting. If you are interested to know and learn more about perfectly secure VSS schemes, you are referred to this survey paper.

Thank you.