


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

Lecture - 22
Properties of Polynomials over a Field: II

Hello, everyone. Welcome to this lecture. So, we will continue our discussion regarding Polynomials over a field.

(Refer Slide Time: 00:29)

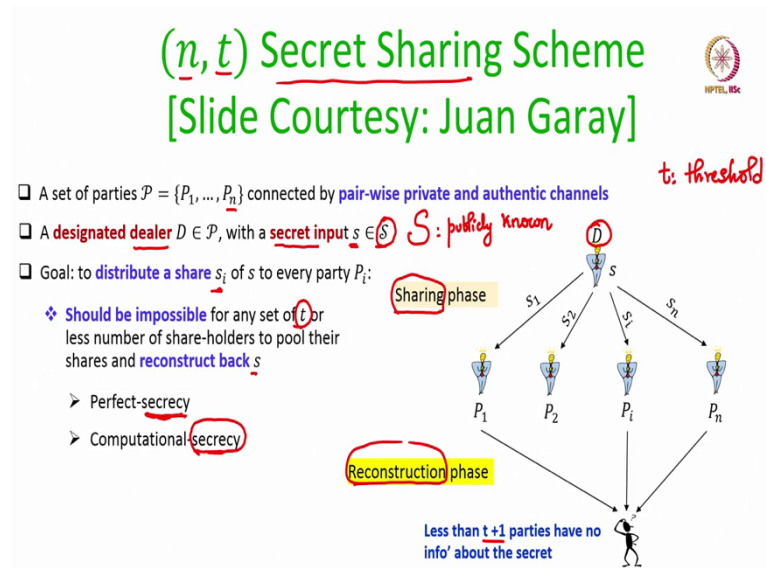
Lecture Overview


IITB, BSC

☐ Relationship between Reed-Solomon (RS) codes and Shamir's secret-sharing

Specifically in this lecture, we will see a very nice relationship between Reed-Solomon codes which we had discussed earlier and another fundamental primitive in distributed cryptography namely Shamir's secret sharing.

(Refer Slide Time: 00:44)



So, let us first try to understand the problem of n, t secret sharing and the slide I have taken from Juan Garay. So, you have two parameters here n and t and what is the setting here? You are given a set of n parties, ok, and imagine that there is a mechanism for secure communication between every pair of parties. And, one of these n parties is a designated party, a dealer, who has some secret input from a bigger space.

So, the secret small s could be as small as a bit or it could be a large element and the set of all candidate secrets is the secret space which is publicly known. So, this secret-space \mathcal{S} is publicly known, but the exact input or the exact value of the secret which dealer has is not known. Now, what is the goal here?

As the name suggests, secret sharing, we want a mechanism which allows the dealer to share his secret by giving some piece of information about the secret to every party and that piece of information is called the share. So, the share for the i -th party is denoted by s_i .

So, we want a mechanism which allows the dealer to distribute his secret and give a share of his secret to every party in such a way that it should be impossible for any subset of t or less number of parties or shareholders to reconstruct back the secret together; that means, if any subset of t shareholders come together and they exchange their shares they tell their shares to each other and then try to compute what exactly was the dealer's secret, they should fail to do that ok.

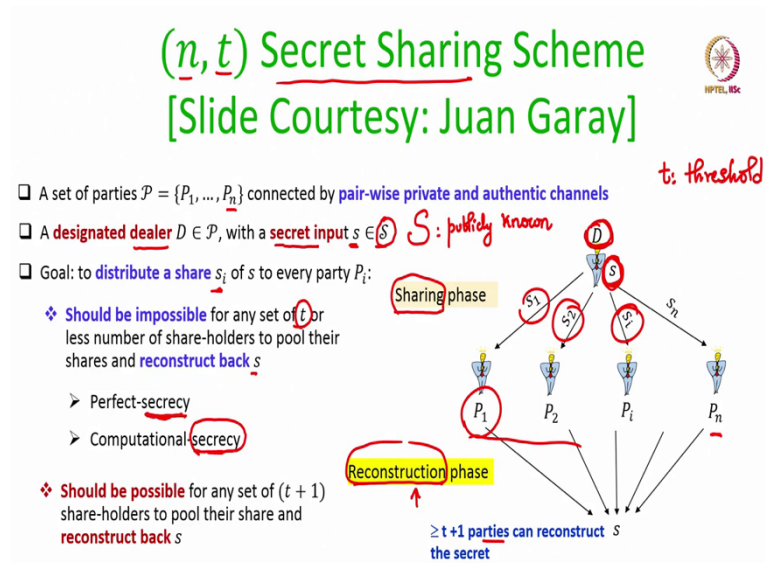
Whereas if $t + 1$ or more shareholders come together and exchange their shares; that means, if $t + 1$ or more number of shares are revealed then the secret can be reconstructed back uniquely. So, t is acting as a threshold here. Up to t number of shares is insufficient to get back the secret, but as soon as $t + 1$ or more number of shares are available the secret can be reconstructed back uniquely.

Now, we can have two flavours of privacy or secrecy here. So, the secrecy property is that any subset of t shareholders should fail to learn the dealer's secret. Now, depending upon whether those t shareholders are computationally bounded or computationally unbounded; that means, whether their computing resources are finite or infinite the notion of secrecy that we achieve is called perfect secrecy or computational secrecy appropriately, ok.

So, if the t shareholders are computationally unbounded and still fail to learn what exactly was the dealer's secret, then that kind of secrecy is called perfect secrecy; whereas, if the t shareholders are computationally bounded and fail to learn the dealer's secret then the notion of secrecy which we achieve is called as the computational or cryptographic accuracy, ok.

And, traditionally this secret sharing scheme will have two protocols – a protocol for the sharing and a protocol for the reconstruction which are basically executed with some gap. What will be the gap? That will be determined later on when we will see the exact the MPC protocols. So, in the sharing phase dealer will share the secret and the requirement will be that any t shareholders if they try to learn the secret before the reconstruction phase, well they will fail to do that.

(Refer Slide Time: 05:06)



And, during the reconstruction phase if $t+1$ or more shareholders come together, if they run the reconstruction algorithm, they should be able to get back the dealer's secret. So, this is a very fundamental primitive in cryptography because there are plenty of real-world scenarios where there might be a dealer with some secret information which it does not want to keep with itself because the dealer might be worried what if the secret gets compromised, gets hacked.

So, instead of storing the secret with itself, it would like to distribute the secret across say n locations by keeping a share of the secret at each of the locations so that later on whenever the secret need to be retrieved any $t+1$ pieces can be pulled and then they can be combined to get back the secret, right.

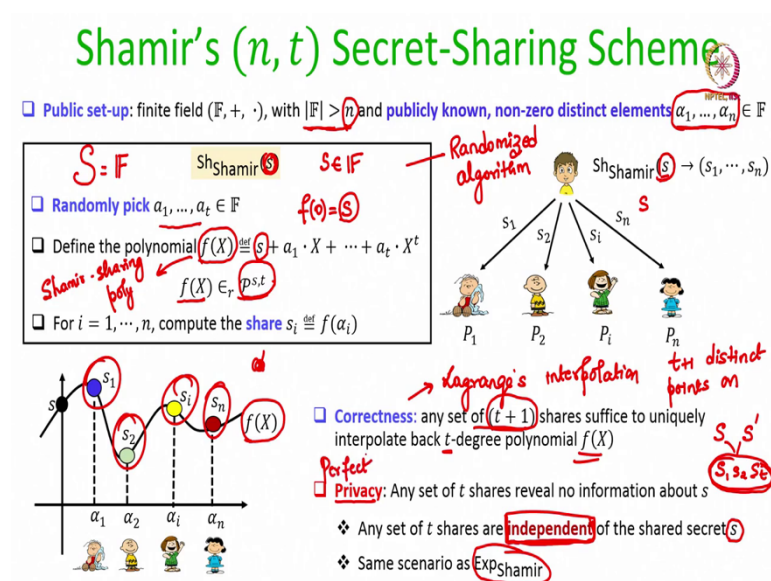
So, the secret could for example, could be the key for your wallet in the bitcoin application, right. So, if that bitcoin wallet key is the secret credential and it is a very sensitive information because if that key is lost then anyone can hack my wallet, so, I would not like to keep it at a single location, but rather I can keep a share at some location, another share at my office, another share at my cupboard and so on.

So, that later whenever indeed I require the key to be retrieved I can go and fetch any t plus 1 shares and then get back the secret. I stress here when I am saying the sharing the secret the shares s_1, s_2, s_i, s_n , they are not copies of s because if the shares are simply copies of s then it

is very easy for any attacker to get back the secret. It just has to go and fetch a copy, that is all.

So, the shares have to be computed in a very clever way, so that up to t shares are insufficient to get back the dealer's secret, but as soon as $t+1$ or more shares are available the secret can be reconstructed back.

(Refer Slide Time: 07:18)



So, there are many instantiations of n, t secret sharing scheme. We will use an instantiation due to Shamir. This is called Shamir's secret sharing scheme. It works over a finite field. So, to instantiate this scheme we will use a finite field. The field could be any finite field. The only restriction is that it should have a size of $n + 1$; that means, it should have at least $n + 1$ elements. So, there will be n publicly known zero nonzero elements from this field which will be available as part of the public setup.

Everyone will be knowing that, whoever are running this instance of Shamir secret sharing. Why we require the field size to be greater than n , because anyhow the element 0 will be there in the field and we require additionally n nonzero elements which have to be distinct. So, that is why the field size has to be at least $n + 1$.

Now, what is the sharing mechanism? To share a secret s where s is an element from the field and that is why the share space, namely the set of all possible secrets, namely this set S is also the field F only. To share an element from the field, the sharing algorithm randomly picks the

coefficients and it forms a t degree polynomial, using those coefficients with the constraint that its constant term is s , namely $f(0)$ will be s and what will be the shares? The shares are computed by evaluating this polynomial at X equal to α_1 , X equal to α_2 , and like that at X equal to α_n .

So, pictorially for instance if this is the dealer and it wants to secret share a secret s , then what it has to do is, basically it has to pick a random t degree polynomial, evaluate it at α_1 and give it as the share to the first party. Evaluate the polynomial at α_2 and give the resultant value as the share to the second party; evaluate the resultant polynomial at α_i and give it as the share to the i -th party and evaluate the polynomial at α_n and give the resultant value to the n -th party as the share.

Well, everyone will know, all the shareholders will know that the i -th share is the evaluation of this polynomial at α_i . The value of α_i will be publicly known because we are assuming here that evaluation points are publicly known. But, what exactly are the contents of the share, what exactly is the value of the share, that will not be known if that party is not corrupt or under the control of the adversary, ok. So, that is a very simple Shamir's sharing mechanism.

Now, we have to see whether it satisfies the two requirements namely that of correctness and privacy. So, let us prove the correctness property, we want to argue here that any set of $t+1$ shares is sufficient to uniquely recover back the secret and that simply comes from the property of Lagrange's interpolation ok.

So, if any $t+1$ shares are available then those $t+1$ shares basically correspond to $t+1$ distinct points on the unknown $f(X)$ polynomial. And, now, we have already discussed that if you are given $t+1$ distinct points on an unknown t degree polynomial you can always uniquely recover back.

And, if you can recover back the polynomial $f(X)$, then the dealer's secret is nothing, but the constant term of that polynomial. So, correctness is trivial. Privacy we want to show that any subset of t shares reveal no information about a secret s , ok. Namely, any subset of t shares, its probability distribution is independent of the secret s .

Now, notice that this Shamir secret sharing is a randomized algorithm. Why it is a randomized algorithm? Because if the dealer wants to share the same secret s multiple times, then each time it will be picking the underlying sharing polynomial, so we will call this polynomial f of X as the Shamir's sharing polynomial Shamir sharing polynomial.

So, the Shamir sharing polynomial is not fixed for a fixed s . For a fixed s , the Shamir sharing polynomial takes different values from this bigger set of all possible t degree polynomials with s being the constant term. That automatically implies that the vector of shares which the dealer will compute for different invocations of the Shamir secret sharing, but for the same input s will be different.

And, that is why there is a probability distribution associated with the output shares because they are probabilistically depending upon the input. So, what we now want to claim here is that if we take any subset of t shares and focus on its probability distribution, it is independent of the dealer's input s . Because if we prove this then that automatically proves the privacy property.

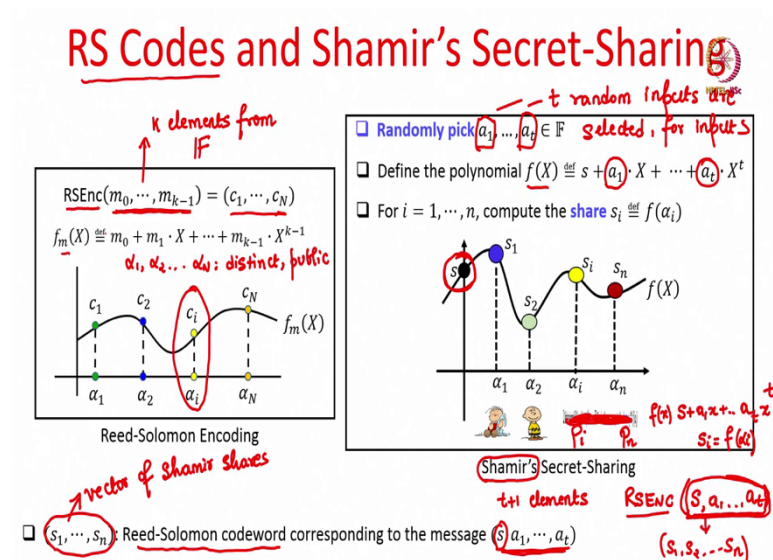
Because if the probability distribution of t shares is independent of the dealer's secret, then it does not matter whether the secret was s or whether the secret was s' , whatever are the t shares say the shares s_1, s_2 and s_t are seen by the adversary or t corrupt parties, the probability distribution of s_1, s_2, \dots, s_t will be independent of whether the secret was s or whether the secret was s prime. And just by observing these t values nothing about the actual input of the dealer can be inferred.

But, this property namely any set of t shares being independent of the shared secret s , it simply follows from the previous experiment which we had seen in the previous lecture, where there was some input and to generate the output for that input, a random sharing polynomial is picked, t degree polynomial is picked and the output is computed as the value of that polynomial at n distinct points.

And, in the context of that experiment we have proved that the probability distribution of t output values is independent of the input and we had actually demonstrated that with an example as well. So, the privacy of the Shamir's secret sharing simply follows from that argument and even if the t shareholders are computationally unbounded, they will fail to infer what exactly was the dealer's secret whether it was s or whether it was s' .

Because with equal probability the t shares which t corrupt shareholders would have seen might come if the dealer's secret would have been s , and with the same probability those t shares might be the output of Shamir's secret sharing if the dealer's secret would have been s' , right. So, that shows the perfect privacy of Shamir's secret sharing.

(Refer Slide Time: 16:02)



Now, we want to see how Shamir's secret sharing can be seen as a special case of Reed-Solomon codes or vice versa. So, let us first recall the Reed-Solomon encoding algorithm. In the context of the Reed-Solomon encoding algorithm there is a message consisting of k elements from the field and to compute the Reed-Solomon code word for this message, where this code word size is big N , a message encoding polynomial is computed that message encoding polynomial is completely determined by the elements of the message.

So, there are k elements, they constitute the k coefficients of the message encoding polynomial and the code word is computed as follows. The i -th component of the code word is the evaluation of this message encoding polynomial at α_i . So, basically this code word is nothing but n distinct values on the message encoding polynomial and here α_1, α_2 and α_n are distinct and publicly known.

Whereas, in Shamir's secret sharing scheme there is only one input, ok. So, the number of inputs in these two primitives is different. In Reed-Solomon encoding, the message input is a

collection of k elements whereas, in the Shamir secret sharing it is only one element which has to be shared but to share it, t dummy elements are randomly picked, ok.

So, t random inputs are selected for input s and then a Shamir-sharing polynomial is fixed based on the selected coefficients. So, it will be a randomly chosen polynomial because the coefficients are randomly chosen except the constant term and the shares are nothing, but the value of that polynomial at α_1 to α_n .

Now, we want to compare these two primitives. Is there any similarity between these two primitives and the answer is yes. If we closely see the vector of Shamir shares, this vector of shares can be interpreted as the Reed-Solomon codeword corresponding to a message which has $t + 1$ elements and where the elements are s, a_1, a_2, \dots, a_t , ok.

Namely, what I am saying is, if the Reed-Solomon encoding algorithm is applied on a message where the message components are s, a_1, \dots, a_t , then it would have resulted in a Reed-Solomon code word s_1, s_2, \dots, s_n of size n . Because the Reed-Solomon encoding algorithm would have selected a t degree polynomial of the form $s + a_1X + \dots + a_tX^t$. And the code word components would have been computed by evaluating this polynomial at α_i which is exactly what the Shamir secret-sharing scheme is doing.

That means, as part of the Shamir secret sharing even though the input size is 1, it is expanded internally into an input of size $t + 1$ by appending t random components and then we can visualize the entire computation of Shamir secret sharing as applying the Reed-Solomon encoding algorithm on the expanded input.

So, you can call this original input of Shamir's secret sharing followed by the concatenated random inputs as the expanded inputs and then we can visualize Shamir's shares as the Reed-Solomon code word corresponding to this expanded message. This property, this relationship between Shamir secret sharing and Reed-Solomon encoding will be later useful when we will use the Shamir secret sharing scheme in several other problems.

So, there we will very frequently use the fact that the vector of shares computed in an instance of Shamir secret sharing scheme can be viewed as a special case of running the Reed-Solomon encoding algorithm.

(Refer Slide Time: 22:11)

References



- ❑ Adi Shamir: How to Share a Secret. Commun. ACM 22(11): 612-613 (1979)
- ❑ Robert J. McEliece, Dilip V. Sarwate: On Sharing Secrets and Reed-Solomon Codes. Commun. ACM 24(9): 583-584 (1981)
- ❑ Ashish Choudhury: Protocols for Reliable and Secure Message Transmission. IACR Cryptology ePrint Archive: 281 (2010)

So, these are the references for today's lecture. The Shamir secret-sharing scheme was introduced in this seminal work and the relationship between Shamir secret-sharing and Reed-Solomon codes was discussed in this paper. And, if you want to know more about secret sharing scheme and their relation to error-correcting codes you can refer to my thesis as well.

Thank you.