


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

**Lecture - 23**  
**One Round PSMT Protocol**

(Refer Slide Time: 00:27)

## Lecture Outline




- ❑ One round PSMT protocol
  - ❖ Protocol from Reed-Solomon (RS) codes

Hello everyone, welcome to this lecture. So, in this lecture we will design a One Round PSMT Protocol based on Reed-Solomon codes.

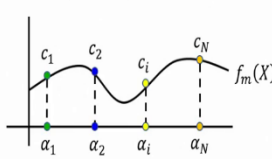
(Refer Slide Time: 00:34)

## RS Codes and Shamir's Secret-Sharing



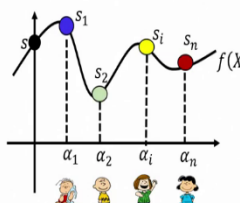
$\text{REnc}(m_0, \dots, m_{k-1}) = (c_1, \dots, c_N)$

$f_m(X) \equiv m_0 + m_1 \cdot X + \dots + m_{k-1} \cdot X^{k-1}$



Reed-Solomon Encoding

- ❑ Randomly pick  $a_1, \dots, a_t \in \mathbb{F}$
- ❑ Define the polynomial  $f(X) \triangleq s + a_1 \cdot X + \dots + a_t \cdot X^t$
- ❑ For  $i = 1, \dots, n$ , compute the share  $s_i \triangleq f(\alpha_i)$



Shamir's Secret-Sharing

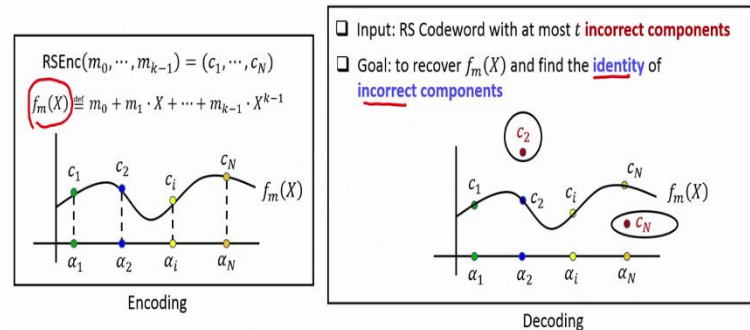
- ❑  $(s_1, \dots, s_n)$  Reed-Solomon codeword corresponding to the message  $(s, a_1, \dots, a_t)$

So, just to recap, in the last lecture we had seen the relationship between Reed-Solomon codes and Shamir's secret-sharing and we have discussed extensively that the vector of shares which is computed as an output during an instance of Shamir's secret sharing can be visualised as a Reed-Solomon code word corresponding to the sender's message being the secret to be shared, followed by the random coefficients which are picked as part of the Shamir's secret sharing scheme.

(Refer Slide Time: 01:18)

## Reed-Solomon Code: Error Correction

$\alpha_1, \dots, \alpha_N$ : publicly-known, **distinct elements** from  $\mathbb{F}$

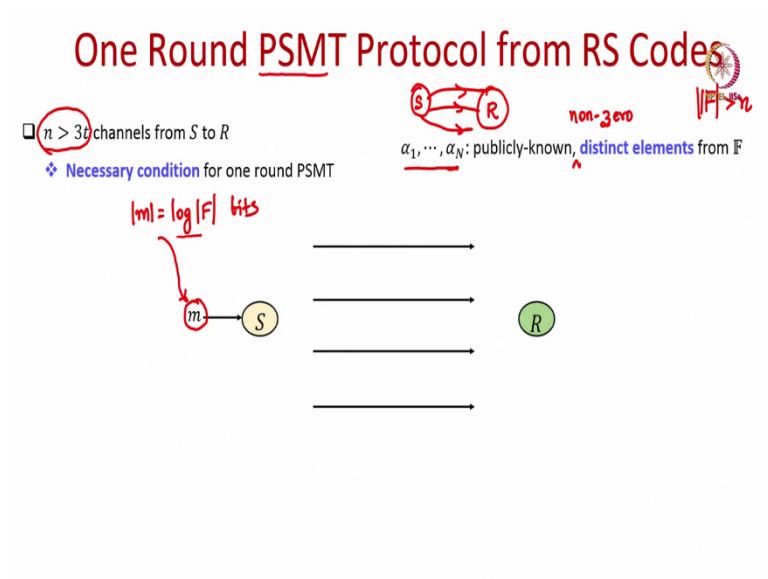


Coding Theory: Decoding possible if and only if  $N - k \geq 2t$

And we had also discussed the Reed-Solomon error correction property in some previous lecture. So, imagine there is a code word corresponding to a message Reed-Solomon code word corresponding to some message which is computed and communicated to the receiver. And suppose some of the components get changed. The goal for the receiver is to recover back the sender's message. For that, it is sufficient to recover back the sender's polynomial even if there are some of the some even if some of the components became corrupt, where the identity of those incorrect components will not be known.

So, that is a decoding problem and we have seen that, from the bounds which are dictated by the coding theory, the decoding is possible if the difference between the size of the code word and a message is at least twice the number of corrupt or incorrect components which the decoding algorithm would like to error correct. And we have also seen a decoding algorithm for the Reed-Solomon codes. Now based on all these things we will see how to design a one round PSMT protocol.

(Refer Slide Time: 02:37)



So, just to recap, PSMT stands for Perfectly Secure Message Transmission problem. And one round means that we have only unidirectional wires from  $S$  to  $R$ ; that means,  $R$  cannot communicate back and we had also discussed that the condition  $n > 3t$  is a necessary condition for any one round PSMT protocol, we have derived this necessary condition.

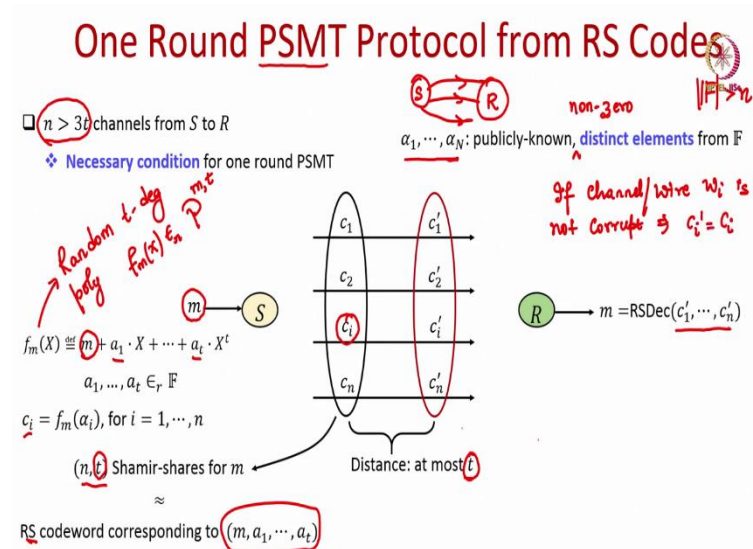
So, assuming you are given at least  $3t + 1$  channels unidirectional channels from  $S$  to  $R$  and at most  $t$  of the  $n$  channels could be byzantine corrupt, we want to design a one round PSMT protocol. So, in this protocol all the operations will be done over a finite field and as a setup, we will assume that there are  $n$  publicly known nonzero distinct elements. So, all the operations will be performed over a finite field where the size of the field is at least  $n + 1$  and sender's message also will be an element of the field.

So, you might be wondering how that is possible, what if sender has a bit string binary string which it wants to securely communicate to receiver? Well, if that is the case then we can always map that binary string as elements of the field and run this one round PSMT protocol. So, for instance if sender has a message consisting of  $\log |\mathbb{F}|$  number of bits then that is binary string can be interpreted as a symbol from the field and then sender can run this protocol.

If the message size is more than  $\log |\mathbb{F}|$  then sender can divide the message into multiple chunks of  $\log |\mathbb{F}|$  bits each string of  $\log |\mathbb{F}|$  bits can be interpreted as an element of the field and then each such field can be securely communicated to the receiver by each such

field element can be securely communicated to the receiver by running this PSMT protocol.

(Refer Slide Time: 05:17)



So, those are some standard mapping issues, we do not want to go into details. We assume here that the sender has some message which is an element of the field, and which is known only to the sender, and we want to design a protocol which allows the sender to do so. Now, what does the sender do? Here is the following. It picks a random  $t$  degree polynomial whose constant term is  $m$  and the remaining coefficients are randomly chosen from the field.

So, basically the polynomial  $f_m(X)$  is a random element from the set of all polynomials of degree  $t$  whose constant term is  $m$  and then sender computes  $n$  distinct points on this polynomial by evaluating this polynomial at  $\alpha_1, \dots, \alpha_n$ . Let the  $i$ th point be  $c_i$  now the sender sends this vector of values  $(c_1, \dots, c_n)$ , where the  $i$ th component is communicated over the  $i$ th channel, to the receiver.

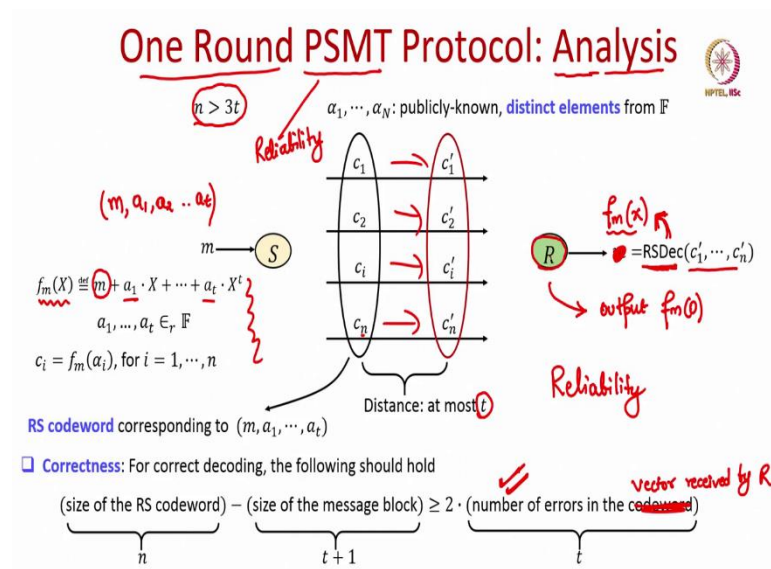
Before going further what exactly has the sender has done? Sender has basically communicated a vector of Shamir shares for the message  $m$  where the degree of sharing is  $t$  such that the  $i$ th share is sent only over the  $i$ th channel. So, the vector is not sent completely over the first channel or the second channel, it is spit across  $n$  channels and over the  $i$ th channel only the  $i$ th share is sent. From our previous lecture we know that Shamir shares also correspond to Reed-Solomon code words.

So, I can also interpret this vector of Shamir shares as the Reed-Solomon code word corresponding to a message block consisting of  $t + 1$  elements  $(m, a_1, a_2, \dots, a_t)$ . Now, suppose receiver receives the components  $c'_1, c'_2, \dots, c'_i, \dots, c'_n$ . Receiver will be knowing that the distance between the received vector and the actual vector which sender has sent is at most  $t$  because it will be knowing that there could be at most  $t$  wires,  $t$  channels, which are corrupt here and which will deliver incorrect shares incorrect component of the Reed-Solomon code words.

But receiver will not be knowing what exactly the identity of those  $t$  corrupt channels,  $t$  corrupt wires, is. And it does not know the sender's original code word as well because sender has sent it over the transmission during the transmission some  $t$  of the components could get changed could get corrupt.

Receiver's goal is to recover back the sender's message. To do that receiver applies the Reed-Solomon decoding algorithm on this received vector. So, let me write out here a fact that if channel wire  $w_i$  is not corrupt then it implies that  $c'_i = c_i$ ; that means, in the received vector at least  $n - t$  components are correct and up to  $t$  components could be incorrect.

(Refer Slide Time: 09:22)



Now, let us see the analysis of this protocol. It is easy to see that it is a one round protocol, because sender computes everything and sends in a single shot to the receiver that is all. So, it is a one round protocol. Let us see whether receiver will be able to recover back the sender's message or not.

So, how is the receiver recovering the sender's message? By applying the Reed-Solomon decoding algorithm on the received vector. Now from the results encoding theory in order that the decoding works correctly for the receiver the following relationship should hold. The size of the Reed-Solomon code word which sender has sent minus the size of the message block of the sender and what is the message block of the sender?

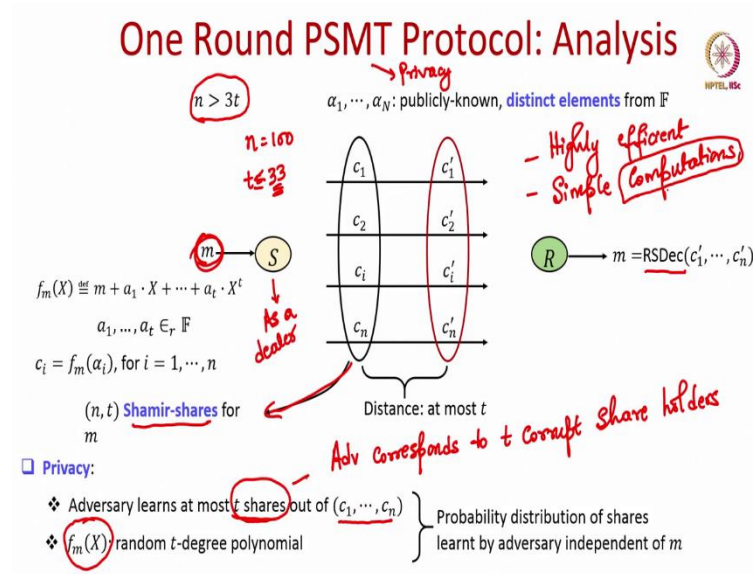
The message block consists of the elements  $(m, a_1, a_2 \dots, a_t)$ . So, it is of size  $t + 1$ . The difference between these two quantities should be greater than equal to 2 times the number of errors which are present in the vector received by receiver. If this relationship holds then from the results in coding theory, we know that Reed-Solomon decoding will correctly output the polynomial  $f_m(X)$ .

So, receiver basically should compute  $f_m(X)$  using the decoding algorithm and once it recovers  $f_m(X)$  it should output  $f_m(0)$  because that is what is the sender's message. But now, we want to analyse what is the guarantee that Reed-Solomon decoding algorithm indeed gives the polynomial  $f_m(X)$  back to the receiver for that this relationship has to be satisfied.

So, now what is the size of the Read-Solomon code word that sender has computed? Its size is  $n$ . And what is the size of the message block of the sender? The size of the message block of the sender is  $t + 1$  because it consists of  $t + 1$  elements. The actual input was  $m$ , but to send it has picked  $t$  dummy components and computed the Reed-Solomon code word. And what is the number of errors which could be present in the vector received by the receiver? At most  $t$  because there could be at most  $t$  corrupt wires.

So, is this relationship true? The answer is yes because we are working with the condition  $n > 3t$ . So, since  $n > 3t$  this condition is guaranteed which ensures that the Reed-Solomon decoding algorithm will recover the polynomial  $f_m(X)$  for the receiver and hence, the receiver will output the correct message.

(Refer Slide Time: 12:55)



So, the correctness is proved. Now, for the reliability part. So, this proof is for the reliability. So, recall that there are two requirements from the PSMT protocol, the first is the reliability property namely the requirement is that irrespective of what adversary does over the  $t$  corrupt wires receiver should be able to output the correct message at the end of the protocol.

And that is what we have proved here we now want to prove the privacy. The second requirement from PSMT. And what is the privacy requirement? The privacy requirement is that whatever information adversary sees over  $t$  wires that should not help the adversary to learn about sender's message. So, how much information can the adversary learn in this protocol? So, imagine adversary sits over the first  $t$  channels or it could be any  $t$  channels, it learns  $t$  shares from this vector of length  $n$ .

Now, what is this vector? This vector is a collection of Shamir shares. And what was the degree of Shamir sharing polynomial? It is a random  $t$  degree polynomial and we have proved in our earlier lecture that in a Shamir secret sharing scheme, if there is an adversary who gets at most  $t$  shares, its probability distribution is independent of dealer's secret. That is what precisely is happening. You can imagine the sender is a dealer and he is sending one share over each channel.

In the worst case,  $t$  shares might get compromised because they might be observed by the adversary. So, adversary corresponds to  $t$  corrupt shareholders and in the context of

Shamir secret sharing, we have proved that even if adversary learns  $t$  shares its probability distribution will be independent of the dealer's secret and dealer's secret here is nothing, but the message for the PSMT and that shows that adversary fails to learn any information about the message even though it is computationally unbounded.

So, that proves the privacy property and you can see that this protocol is highly efficient right. Why it is highly efficient? Because it requires only simple computations what are the computations required here? At the sender's end it requires computing Shamir shares and at the receiver's end it requires running an instance of decoding algorithm and all are all of these are field operations finite field operations and there are very fast algorithms standard fast algorithms to perform finite field operations and that is why the computations which are performed here are very efficient.


So, that is why this protocol can be deployed in practice without any difficulty. So, it would not be a computational bottleneck for you, and it gives you a very strong security guarantee in the sense that even if the adversary is computationally unbounded it still cannot figure out what exactly was sender's message. The only disadvantage that you might say here regarding the protocol is that it requires a lot of channels lot of wires between  $S$  and  $R$ .

So, for instance if I take  $n = 100$  then this protocol will work as long as up to 33 wires are compromised. So, even if 33 wires get compromised, if the remaining 67 wires are not corrupt, this protocol will work. But that requires a very significant proportion of honest wires compared to the corrupt wires, but with that price you are actually getting very nice guarantee security guarantees, namely perfect security, and that too with high efficiency.



(Refer Slide Time: 18:02)

## References



- ❑ D. Dolev, C. Dwork, O. Waarts and M. Yung: Perfectly Secure Message Transmission. JACM 40(1): 17-47, 1993
- ❑ Ashish Choudhury: Protocols for Reliable and Secure Message Transmission. IACR Cryptology ePrint Archive: 281 (2010) ✓

So, the one round protocol that I had discussed in today's lecture was presented in the seminal work by Dolev et al which introduced the PSMT problem and if you want to know more about the PRMT and PSMT protocols you can refer to this thesis.

Thank you.