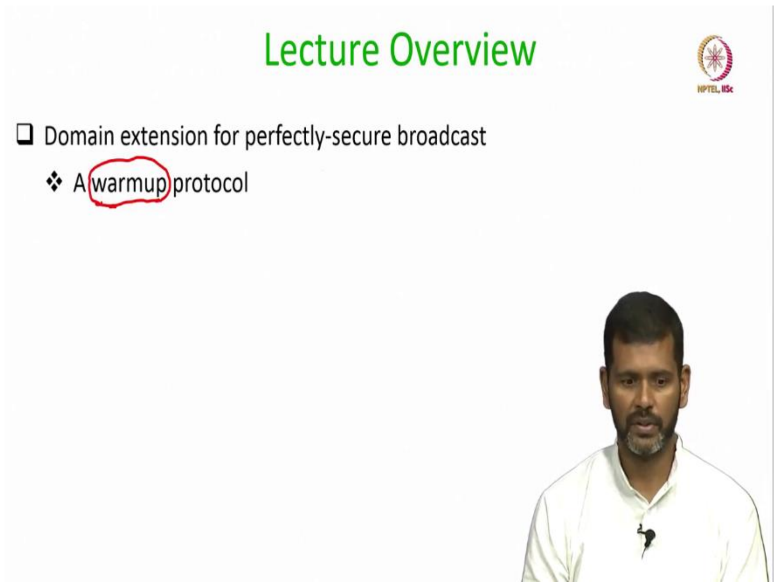
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 27
Domain Extension for Perfectly-Secure Broadcast Based on RS Error-Correcting Codes: II

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:24)

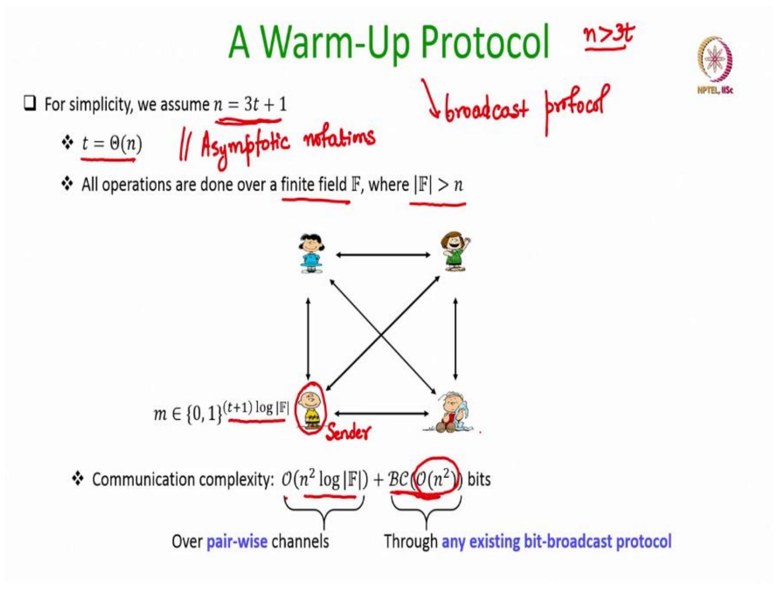


Lecture Overview

- ❑ Domain extension for perfectly-secure broadcast
  - ❖ A warmup protocol

So, now, in this lecture we will continue our discussion regarding domain extension for perfectly secure broadcast. And we will first see a warm-up protocol this will not be the exact domain extension protocol, and this will be inefficient as well, but later on we will see how this form of protocol is used in the domain extension protocol.

(Refer Slide Time: 00:46)



A Warm-Up Protocol

$n>3t$

- For simplicity, we assume $n = 3t + 1$ ↓broadcast protocol
  - $t = \Theta(n)$ // Asymptotic notations
  - All operations are done over a finite field $\mathbb{F}$, where $|\mathbb{F}| > n$

$m \in \{0,1\}^{(t+1)\log|\mathbb{F}|}$
Sender

- Communication complexity: $\underbrace{O(n^2 \log|\mathbb{F}|)}_{\text{Over pair-wise channels}} + \underbrace{\mathcal{BC}(O(n^2))}_{\text{Through any existing bit-broadcast protocol}}$ bits

So, in this warm-up protocol for simplicity we will assume $n = 3t + 1$. So, remember that for perfectly secure broadcast and Byzantine agreement and the condition $n > 3t$ is necessary and sufficient. So, $n = 3t + 1$ is the minimum value of n for any given t satisfying the condition $n > 3t$.

The reason we are making this assumption $n = 3t + 1$ is that it helps us during the analysis of the communication complexity because, we can then use the fact that asymptotically $t = \theta(n)$. So, I am assuming here that all of you are familiar with big-O, Ω and θ notations; they are asymptotic notations and if you are not familiar with them please refer to any standard text on data structures and algorithm.

Also in this warm up protocol we will assume that all the operations are performed over a finite field $\mathbb{F}$, where the only restriction on the field is that it should be of size at least $n + 1$. Now, in this warm-up protocol the sender will have a message m which will be of size $(t + 1) \cdot |\mathbb{F}|$ bits and it will be a broadcast protocol.

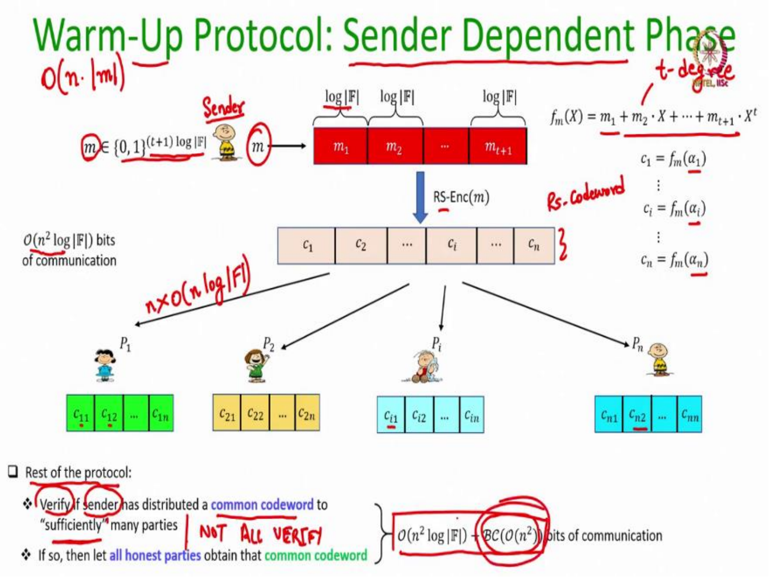
So, this is a broadcast protocol, where the goal of the sender will be to send its message identically to all the parties and that should happen even if the sender is potentially corrupt. And the overall communication complexity of the warm-up protocol will turn out to be this much.

So, $n^2 \log|\mathbb{F}|$ communication will happen over the pair wise channels among the parties and apart from that the protocol will also incur a communication will also incur a broadcast of n^2 bits and that broadcast can be instantiate using any existing bit broadcast protocols.

So, that means, there are two parts of the communication complexity in this form of protocol whatever communication happens over the point-to-point channels and whatever communication should happen by invoking instances of bit broadcast. So, the communication, which should happen through instances of bit broadcast that will be of order n^2.

(Refer Slide Time: 03:43)

Warm-Up Protocol: Sender Dependent Phase



$O(n \cdot |m|)$
Sender
$m \in \{0,1\}^{(t+1)\log|\mathbb{F}|}$
m
$\log|\mathbb{F}|$
$\log|\mathbb{F}|$
$\log|\mathbb{F}|$
m_1 m_2 ⋯ m_{t+1}
t-degree
$f_m(X) = m_1 + m_2 \cdot X + \cdots + m_{t+1} \cdot X^t$
$c_1 = f_m(\alpha_1)$
$c_i = f_m(\alpha_i)$
$c_n = f_m(\alpha_n)$
RS-Enc(m)
RS-Codeword
c_1 c_2 ⋯ c_i ⋯ c_n
$O(n^2 \log|\mathbb{F}|)$ bits of communication
$n \times O(n \log|\mathbb{F}|)$
P_1 P_2 P_i P_n
c_{11} c_{12} ⋯ c_{1n}
c_{21} c_{22} ⋯ c_{2n}
c_{i1} c_{i2} ⋯ c_{in}
c_{n1} c_{n2} ⋯ c_{nn}
NOT ALL VERIFY

- Rest of the protocol:
  - Verify if sender has distributed a common codeword to "sufficiently" many parties
  - If so, then let all honest parties obtain that common codeword

} $O(n^2 \log|\mathbb{F}|) + BC(O(n^2))$ bits of communication

So, there will be several phases in the protocol the first phase will be a sender dependent phase, which will depend completely upon the sender because sender will have to send these messages. So, remember recall that sender's input, is a message consisting of $(t+1) \cdot |\mathbb{F}|$ bits.

So, what the sender does is the following, it divides its message into several blocks of $\log|\mathbb{F}|$ bits and each such block of $\log|\mathbb{F}|$ bits can be interpreted as an element of the field, because each element of the field can be represented by $\log|\mathbb{F}|$ bits. So, whatever is the first chunk of $\log|\mathbb{F}|$ bits that is converted into a field element let us denote it by m_1.

In the same way next block of $\log|\mathbb{F}|$ bits of this message m will be interpreted as a field element. We denote it by m_2 and like that the last chunk of $\log|\mathbb{F}|$ bits for of the message small m will be mapped to the field element m_{t+1}. This is what sender is doing. Now the goal of the sender is to send this message identically to everyone.

So, if we would have been in the semi honest setting where all the parties including the sender also would have followed the protocol instructions correctly. Then sender could have simply sent his message over the point-to-point channel to every other party. By the way we are assuming here that there is a pair wise secure channel between every pair of parties namely we are in the secured channel model.

So, if it if you would have been guaranteed that sender will not misbehave, then the simple way of broadcasting senders message will be to ask the sender to send its message individually to the first party, individually to the second party, individually to the nth party that would have require a communication complexity of order of n times whatever is the size of the message, but it is not guaranteed that sender is honest sender could misbehave it can send different m to P_1 and different m to P_2 and so on.

So, that is why to send its message what the sender does first here is the following it computes a Reed Solomon codeword corresponding to its message. So, notice that even though the sender's message is a bit string, it can be mapped into a message consisting of t field elements and then we can compute a Reed Solomon codeword corresponding to that message namely the sender will form a message encoding polynomial whose degree will be t and there will be n publicly known evaluation points $\alpha_1, \alpha_2, \dots, \alpha_i, \dots, \alpha_n$.

And that message encoding polynomial will be evaluated at $\alpha_1, \alpha_2, \dots, \alpha_i, \dots, \alpha_n$ to get the corresponding Reed Solomon codeword whose components are $c_1, c_2, \dots, c_i, \dots, c_n$. Now to send the message m sender sends this full codeword vector to every party. So, to the first party it sends the codeword to the second party, it sends the codeword to the ith party, it sends the codeword to the nth party it sends the code word.

Now, if sender is honest, it would have sent the same codeword copied to every other party, but it is not guaranteed we do not know whether sender is honest or not. So, that is why the codeword copy which P_1 receives there I am using 2 level of indexes indices. So, $P_1{}'$s codeword will be denoted by$c_{11}, c_{12}, \dots, c_{1n}$. So, the second level of indexing denotes the codeword component and the first level of indexing denotes the copy of the sender's codeword as received by P_1.

In the same way I am using a 2-level indexing for the copy of the codeword which P_2 has received, a 2-level indexing for the copy of the codeword which P_i has received and a 2-level indexing for the code what copy which P_n has received from the sender. How much communication this will require? Well, to send the codeword to one party it will require a communication of order of $n \log|\mathbb{F}|$ bits because, the code word consists of n field elements and each field element is represented by $\log|\mathbb{F}|$ bits.

And since sender is sending the codeword to every party the overall cost will be n times the cost of sending the codeword to one party and that is why the cost of this phase will be order of $n^2 \log|\mathbb{F}|$ bits. Notice that, no broadcast has happened as of now; that means, no invocation of bit broadcast has happened to be more specific because, this communication is happening over only the point to point channels.

Now one sender has communicated its copy of the codeword to the respective party, the rest of the protocol will do the following. In the rest of the protocol the parties will interact, and they will try to verify whether sender is behaving honestly and whether sender has distributed a common codeword to sufficiently many parties. Ideally, this should be the case if sender is honest because, sender should have sent the same common codeword to all the parties, but as I said if the sender is Byzantine corrupted, it may not do so.

So that is why now the parties have to interact and verify whether the sender has distributed a common codeword to sufficiently many parties. I stress sufficiently many parties and not all parties because; we can never verify whether the sender has given the same codeword to all the parties.

Because if say P_n is corrupt, it may simply say that I have received the junk from the sender, it can simply say it can simply produce a junk copy of the codeword claiming that it has received from the sender even though sender has not sent it and in that case we cannot trust whether the sender has indeed cheated P_n by sending a junk codeword or whether it is P_n who is reporting incorrectly against the sender. So, that is why we can only guarantee here that all the honest parties can verify whether they have received the same common codeword or not.

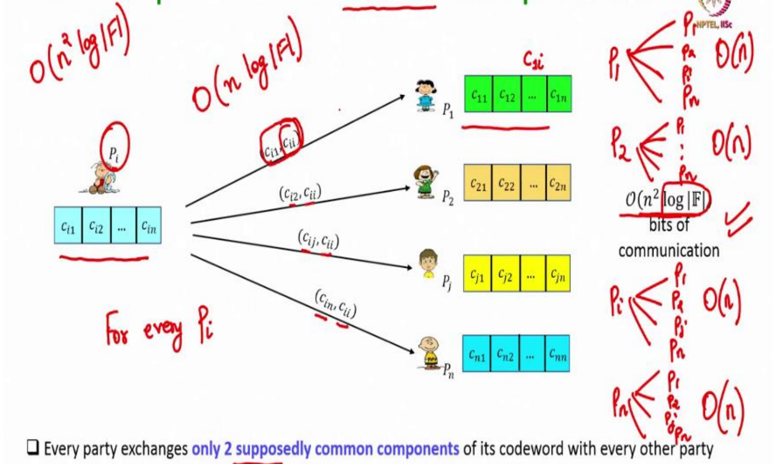
So, that is why this sufficiently right now is in quote unquote, but basically, we will try to ensure that sufficiently many honest parties confirm whether they have received a common codeword from the sender or not. And if it is guaranteed that sender has distributed a common codeword to sufficiently many honest parties, then the next goal will be to ensure that all honest parties, apart from those sufficiently many parties also obtain that common code word.

If that is ensured, then it will be now guaranteed that in the system all honest parties have a common codeword received from the sender and the message corresponding to that codeword will be the overall output for the warm up protocol. Now for this rest of the protocol part, where the parties need to do the verification and then they need to the transfer of the common code word, the allowed communication budget for us is this much.

Namely, the total communication of the over the point to point channel should be $n^2 \log|\mathbb{F}|$ and if at all we are invoking any instance of the bit broadcast there can be at most order of n^2 such instances that is what is allowed, because that is what we require from this warm up protocol when we plug it later in the domain extension protocol.

(Refer Slide Time: 12:23)

Warm-Up Protocol: Sender Independent Phase



$O(n^2 \log|\mathbb{F}|)$
$O(n \log|\mathbb{F}|)$
P_i
c_{i1} c_{i2} ... c_{in}
For every P_i
(c_{i1}, c_{ii})
(c_{i2}, c_{ii})
(c_{ij}, c_{ii})
(c_{in}, c_{ii})
c_{1i}
P_1 c_{11} c_{12} ... c_{1n}
P_2 c_{21} c_{22} ... c_{2n}
P_j c_{j1} c_{j2} ... c_{jn}
P_n c_{n1} c_{n2} ... c_{nn}
P_1 $O(n)$
P_2 $O(n)$
$O(n^2 \log|\mathbb{F}|)$ bits of communication
P_i $O(n)$
P_n $O(n)$

❑ Every party exchanges only 2 supposedly common components of its codeword with every other party

So, now, let us see how the verification proceeds. This will be now be a sender independent phase and this is this happens once the sender has distributed the code words to the respective parties. So, let us see what the; what is the interaction happen what kind of interaction happen during the verification process?

So, recall that party P_i has its own version of the local codeword as it has received from the sender and every other party has its own version of the local code word. One simple way of verifying whether sender has given the common codeword to sufficiently many parties is that, ask every party to broadcast their received common codeword by invoking instances of the bit broadcast.

That will be simply requiring an enormous amount of communication that will not satisfy our allowed budget. So, we have to do the verification in a communication saving way. So, to do the verification every pair of parties exchange only two supposedly common components of their respective code words. So, for instance, if sender would have been honest then we expect that the first component of the ith party's codeword and the first component of the first party's codeword should be identical.

And the ith component of the ith party's codeword should be same as the ith component of the first party's code word. So, that is what party P_i tries to verify. So, party P_i sends only two common components it is not sending the full vector full codeword vector to P_1 and asking P_1 to check whether the codeword which P_i has received from sender is same as the codeword which P_1 has received from the sender.

No. They are only trying to verify the commonality of 2 supposedly common components. In the same way with the second party P_i exchange exchanges only the second component of its codeword and ith component of its codeword with the jth party it exchanges the jth component of its codeword and ith component of its own codeword. And with the nth party it exchanges the nth component of its code word and ith component of its codeword.

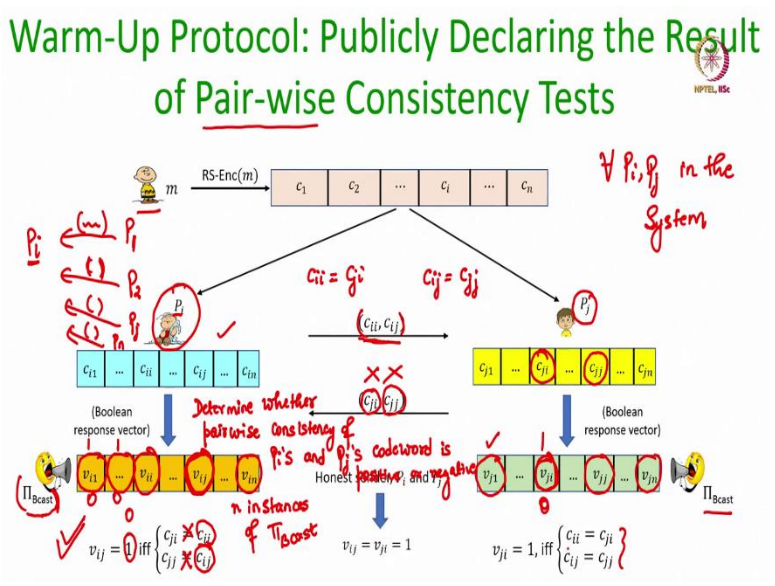
How much communication this phase will require? This will require a total communication of $\theta(n^2 \log|\mathbb{F}|)$ bits because, for one party only this will require $\theta(n \log|\mathbb{F}|)$ bits of communication, but this pair wise exchange happens for every P_i. So, I have demonstrated it only for 1 P_i, but this pair wise exchange of two supposedly common point is happening for every party.

So, P_1 would have been exchanging two common components with $P_1, P_2, \dots, P_i, .., P_n$. P_2 also would have been exchanging two common components on its received codeword with every other party, and like that P_i is anyhow exchanging two common components of its received codeword with every other party, and like that P_n also would have been exchanging two common components on its received codeword with every other party.

So, $\theta(n)$ field elements for 1 party, $\theta(n)$ field elements for the second party, $\theta(n)$ field elements for the i th party and $\theta(n)$ field elements for the nth party. So, that is why total it will be $\theta(n^2)$ field elements of communication and each field element is represented by $\log|\mathbb{F}|$ number of bits. So, that is why this phase will incur a communication complexity of this much these many bits.

So, we have not crossed our allowed budget; that means, till now the sender dependent phase communication complexity was n^2 field elements and so, is the communication complexity for this phase as well. Until now no instance sub bit broadcast has been invoked.

(Refer Slide Time: 16:59)


Warm-Up Protocol: Publicly Declaring the Result of Pair-wise Consistency Tests
NPTEL, IISc
RS-Enc(m)
m
c_1 c_2 … c_i … c_n
$\forall P_i, P_j$ in the System
$c_{ii} = c_i$
$c_{ij} = c_j$
(c_{ii}, c_{ij})
(c_{ji}, c_{jj})
P_i
P_j
c_{i1} … c_{ii} … c_{ij} … c_{in}
c_{j1} … c_{ji} … c_{jj} … c_{jn}
(Boolean response vector)
Determine whether pairwise consistency of P_i's and P_j's codeword is positive or negative
v_{i1} … v_{ii} … v_{ij} … v_{in}
v_{j1} … v_{ji} … v_{jj} … v_{jn}
Honest P_i and P_j
Π_{Bcast}
n instances of Π_{Bcast}
$v_{ij} = 1$ iff $\begin{cases} c_{ji} = c_{ii} \\ c_{jj} = c_{ij} \end{cases}$
$v_{ij} = v_{ji} = 1$
$v_{ji} = 1$, iff $\begin{cases} c_{ii} = c_{ji} \\ c_{ij} = c_{jj} \end{cases}$

Now based on the pair wise consistency test, which happens here, every party will now publicly declare the result whether the test is positive or negative. So, what does that mean? So, recall that till now the following has happened sender had a message and it computed Reed Solomon codeword for that message a copy of that codeword was sent to the ith party and a copy of that codeword would have been sent to the jth party for all (P_i, P_j) in the system.

And then after that P_i would have sent two common values two supposedly common values on its copy of the received codeword to P_j and P_j also would have sent two supposedly common value on its received codeword to P_i. Ideally if sender P_i and P_j are honest then component wise c_{ii} should have been same as c_{ji} and c_{ij} should have been same as c_{jj}. So, that is what should have happened. If both if sender P_i and P_j are all honest, but we do not know whether sender P_i and P_j are all honest or not they may or may not be.

So, now what every party does is the following it prepares a Boolean response vector based on the values, which it receives from the other parties and P_j also will prepare a Boolean response vector. Now if we consider the ith party's Boolean vector the jth component of that vector determines whether pair wise consistency of P_i's and P_j's codeword is positive or negative.

Namely P_i will say that ok I am fine with P_j, which is equivalent to saying that I mark P_j as 1 namely my response for j is 1. Only if the c_{ji} component received from P_j matches my own ith component and the c_{jj} component received from P_j matches my jth component, which should ideally hold if sender P_i and P_j are all honest.

But if either this condition is not satisfied or this condition is not satisfied then P_i will conclude that its either the sender who is faulty and has sent different code words to me and P_j or its P_j who is corrupt and he is unnecessarily exchanging incorrect components with me due to which the checking fails, but P_i cannot conclude whether it is a sender who is corrupt or whether it is a P_j who is corrupt. He will simply accuse P_j that ok no, I am not matching with j so, that is my response for j is 0.

So, this that is the way P_i would have prepared the Boolean response vector. So, P_i would have received 2 values from P_1 from P_2 it would have received 2 values from P_j it would have received 2 values and from P_n it would have received 2 values. The pair of values received from P_1 is compared by P y and if the pair wise consistency is satisfied then he will mark P_1 to be ok; otherwise, he will mark P_1 to be not ok.

In the same way the pair of values received from P_2 is verified by P_i against its own codeword and accordingly P_2 is marked as 1 or P_2 is marked as 0 and so on. In the same way party P_j it is also receiving a pair of values from P_i and the way it marks its ith entry namely the way it labels the ith party is as follows, it checks whether these two conditions are satisfied if they are satisfied then he says that, I am ok with 1, otherwise you will say I am not ok with i; it will say I am ok with i otherwise it will say I am not ok with i.

And like that P_j would have filled each of the response vector bits. Now once every party has prepared its Boolean response vector it makes it public and when I say it makes it public to do that, it invokes an instance of existing bit broadcast protocol. So, now P_i acts as a sender, where the sender input is this Boolean vector at in it invokes instances of any bit broadcast protocol say the king face protocol to broadcast the first bit in its response vector, the second bit in its response vector, the ith bit in its response vector, the jth bit in its response vector and the nth bit in its response vector.

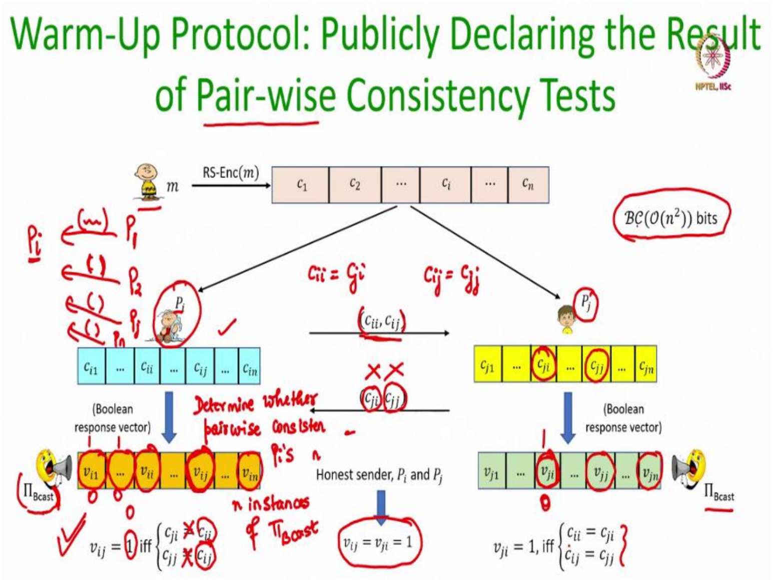
So, namely n instances of bit broadcast. So, that all the parties receive an identical copy of ith party's bit ith party's Boolean vector.

In the same way P_j also will invoke instances of any existing bit broadcast protocol by acting as a sender to broadcast the individual bits in its response vector. Now see here if P_i for instance if its corrupt then instead of he can first fill its Boolean vector arbitrarily; that means, even if the Boolean vector entries should be 1 it can simply say it's 0; that means, it may simply unnecessarily accuse P_j or it may simply broadcast a garbage Boolean vector, which has no meaning at all.

So, we cannot ensure that whether every party is broadcasting its genuine Boolean vector, but what we know is that if P_i is honest if P_i is honest then it will indeed fill its Boolean vector as per this assigned process and indeed it will broadcast its Boolean vector correctly.

So, honest parties they are guaranteed to compute their Boolean vectors as per the protocol steps and they have guaranteed to broadcast the genuine Boolean vector. And as I said earlier if the sender is honest, party P_i is honest and party P_j is honest then the jth component in ith party's Boolean vector and the ith component in the jth party's Boolean vector should be 1; that means, they should be equal.

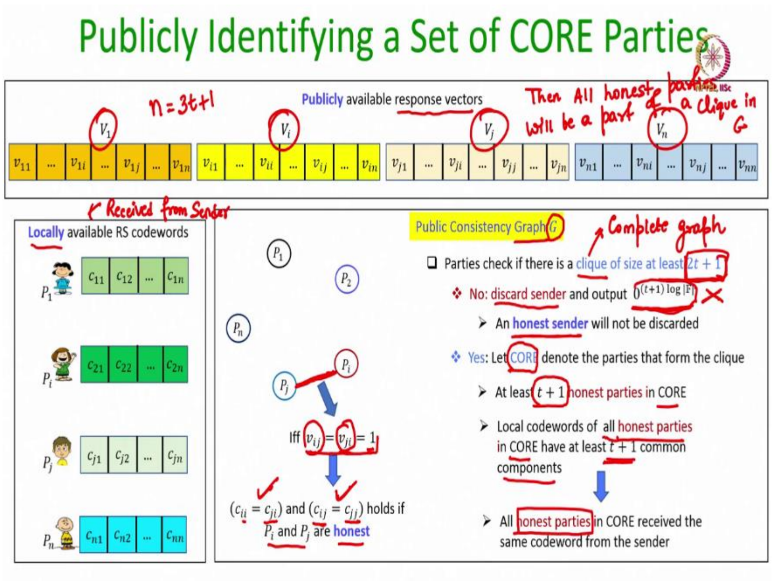
(Refer Slide Time: 25:18)


Warm-Up Protocol: Publicly Declaring the Result of Pair-wise Consistency Tests

NPTEL, IISc
m
RS-Enc(m)
c_1 c_2 … c_i … c_n
$\mathcal{BC}(O(n^2))$ bits
$c_{ii} = c_{ji}$
$c_{ij} = c_{jj}$
P_i
P_j
(c_{ii}, c_{ij})
(c_{ji}, c_{jj})
c_{i1} … c_{ii} … c_{ij} … c_{in}
c_{j1} … c_{ji} … c_{jj} … c_{jn}
(Boolean response vector)
Determine whether pairwise consistent P_i's
v_{i1} … v_{ii} … v_{ij} … v_{in}
v_{j1} … v_{ji} … v_{jj} … v_{jn}
Π_{Bcast}
Honest sender, P_i and P_j
$v_{ij} = v_{ji} = 1$
n instances of Π_{Bcast}
$v_{ij} = 1$ iff $\begin{cases} c_{ji} = c_{ii} \\ c_{jj} = c_{ij} \end{cases}$
$v_{ji} = 1$, iff $\begin{cases} c_{ii} = c_{ji} \\ c_{ij} = c_{jj} \end{cases}$

And this phase requires instances of any existing bit broadcast protocol specifically there are n^2 instances of existing bit broadcast protocol and that will amount to a total broadcast of n^2 bits, which is within our allowed budget of the warm-up protocol.

(Refer Slide Time: 25:38)


Publicly Identifying a Set of CORE Parties

Public Consistency Graph G

- Parties check if there is a clique of size at least $2t + 1$
  - No: discard sender and output $0^{(t+1)\log|\mathbb{F}|}$
    - An honest sender will not be discarded
  - Yes: Let CORE denote the parties that form the clique
    - At least $t + 1$ honest parties in CORE
    - Local codewords of all honest parties in CORE have at least $t + 1$ common components
    - All honest parties in CORE received the same codeword from the sender

NPTEL, IISc
$n = 3t+1$
Publicly available response vectors
Then All honest parties will be a part of a clique in G
V_1 V_i V_j V_n
v_{11} … v_{1i} … v_{1j} … v_{1n}
v_{i1} … v_{ii} … v_{ij} … v_{in}
v_{j1} … v_{ji} … v_{jj} … v_{jn}
v_{n1} … v_{ni} … v_{nj} … v_{nn}
Received from Sender
Locally available RS codewords
P_1 c_{11} c_{12} … c_{1n}
P_i c_{21} c_{22} … c_{2n}
P_j c_{j1} c_{j2} … c_{jn}
P_n c_{n1} c_{n2} … c_{nn}
Complete graph
P_1 P_2 P_n P_i P_j
Iff $v_{ij} = v_{ji} = 1$
$(c_{ii} = c_{ji})$ and $(c_{ij} = c_{jj})$ holds if P_i and P_j are honest

Now, after the parties make public their response vector they will be known to everyone and notice that till now the following communication has happened. So, parties have made their response vectors public and each party has its own copy of the Reed Solomon codeword received from the sender. Now based on the publicly available response vectors the parties will construct a graph, which is called a consistency graph.

And where the nodes of the graph will be the identity of the parties namely P_1, P_2, P_j and P_n and an edge will be added between the parties P_i and P_j if and only if the jth component in the ith party response vector and the ith component in the jth party's response vector are 1; that means, the party P_i and P_j are mutually consistent with each other. So, if this condition is satisfied, we will say that ith party and the jth party are payer wise consistent with each other.

And this means that if P and P_j are honest then component wise in the ith party's codeword the ith component is same as the ith component in the jth part is codeword. And in the same way component wise the jth component in the jth party's codeword is same as the jth component in the ith party's codeword because, that is why party P_i and P_j would have broadcasted $v_{ij} = 1$ and $v_{ji} = 1$ if both of them are honest because, they would have verified that these two conditions are satisfied.

Of course, if P_i and P_j are not honest; that means, if even if these conditions are not satisfied it may happen that the corrupt party among P_i and P_j may unnecessarily say I am ok with the other party that is fine. In that case, we cannot claim anything, but what we can claim is that if at all there is an edge between a pair of honest $(P_i\ , P_j)$ in the graph that means, component wise their code words their respective local code words have the same two components.

Namely, the ith component in their respective local code words are same and the jth component in their respective local code words are also same. Now parties check if there is a clique and when I say clique, I mean to say a complete graph, they check if there is a clique or a complete graph of size at least $2t + 1$ present in the consistency graph. And if they see that there is no such clique present then they discard the sender.

When I say discard the sender, I mean to say stop the protocol at that point only do not do anything else and simply output bit string consisting of $t + 1 \log |\mathbb{F}|$ number of 0's as the overall output for the protocol. So, first of all notice that this graph will be public; that means, all the parties will be constructing the same copy of the conflict graph, this is because the conflict graph is computed based on the response vectors, which have been made public using invocations of the bit broadcast.

And the bit broadcast would have ensured its consistency property would have ensured that every honest party receives the same version of V, same version of V_i, same version of the vector V_j same vector version of the vector V_n; that means, if party 1 has added the edge P_i - P_j in its copy of the consistency graph.

And if P_1 is honest then it is guaranteed that every other honest party also would have added the same edge. Now what is the logic behind discarding the sender here? If sender is honest then all honest parties will be a part of a clique in the graph G.

Because every honest party P_i will be consistent with every other honest party P_j because, an honest sender would have sent the same common codeword to all the honest parties even to in fact, to every other party, but corrupt party might simply replace those code words with some other junk codeword and might proceed in the protocol, but all the honest parties will stick to the version of the codeword as sent by the honest sender.

And when every pair of honest parties P_i and P_j would have exchanged the common components, they will see that they are supposedly common components are indeed common and that is why they would have marked the other party as 1 in their respective response vectors. So, if sender is honest, we expect that all honest parties are a part of a clique in the graph and how many such honest parties are guaranteed at least $2t + 1$.

Because we are working with the set in the setting where $n = 3t + 1$. So, at most t parties can be corrupt who may not be pairwise consistent with the honest parties, but there are at least $2t + 1$ honest parties who are guaranteed to be pairwise consistent and they will be a part of the clique.

So, for an honest sender this step will not be executed because this condition will not be satisfied that implies that if at all a clique of size $2t + 1$ is not found in the graph; that means, the sender is definitely corrupt, and it has not distributed the common codeword a common codeword to all the honest parties.

So, it is fine to simply stop the protocol assuming pretending the dealer to be corrupt and take some default output as its message; that means, we do not care what exactly was senders message. Since you have not behaved honestly during the protocol execution, we will not participate further and we will treat as if you wanted to broadcast a message consisting of all zeros that is, the logic behind this step of discarding the data.

However, it could be possible that a clique of size $2t + 1$ is obtained even if the sender is corrupt of course, for the honest sender a such a clique is guaranteed but it could be possible that even a corrupt deal even a corrupt sender sorry not dealer even a corrupt sender has sent the same common codeword to sufficiently many honest parties and a clique of size $2t + 1$ is obtained.

If a clique of size $2t + 1$ is obtained let us call that subset of parties as CORE. Now what we can conclude regarding code? Well, we can conclude that there are at least $t + 1$ honest parties in code even though its cardinality is $2t + 1$, t of those $2t + 1$ parties may be corrupt under adversaries control in the worst case, but even in that case there are at least $t + 1$ honest parties in the CORE who are guaranteed to be honest.

And those honest parties would have genuinely performed a verification pair wise consistency test and would have marked the other parties other honest parties within the CORE as 1 in their respective response factors; that means, the local code words of all honest parties.

I stress only the honest parties in CORE I am not saying all parties in CORE that is I am stressing all honest parties in CORE have at least $t + 1$ common components because, that is why they have constituted a clique in the graph because, whenever there is an edge between a pair of honest notes in the consistency graph; that means, they have common components they have respective common components in their local Reed Solomon code words.

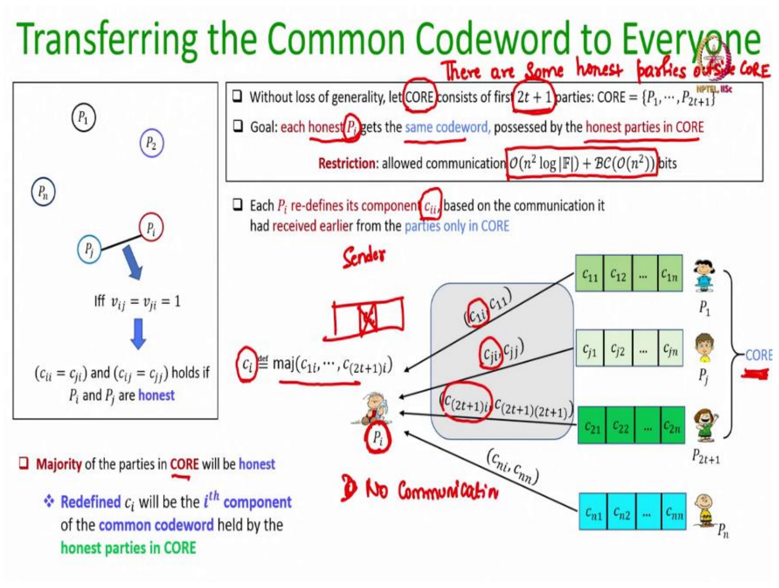
And how many such edges are there between pairs of honest parties? We are guaranteed that there is a clique involving at least $t + 1$ honest parties. So, based on their response we can conclude that there are $t + 1$ honest parties in the CORE and their respective local code words have at least $t + 1$ common components.

Now here we trigger 1 of the properties which we have discussed regarding Reed Solomon code words in our earlier lecture, which says that if you have 2 code words which have

$t + 1$ or more common components corresponding to message encoding polynomials of degree t, then that implies that the 2 code words are the same code words.

So, what is the message encoding polynomial? In this case the message encoding polynomial its degree is t and we have come to the conclusion that the local code words of all honest parties in CORE have $t + 1$ or more $t + 1$ or more number of common components that automatically implies that all honest parties in the CORE have received the same codeword from the sender.

(Refer Slide Time: 36:09)



Transferring the Common Codeword to Everyone

There are Some honest parties outside CORE

- Without loss of generality, let CORE consists of first $2t + 1$ parties: CORE = $\{P_1, \cdots, P_{2t+1}\}$
- Goal: each honest P_i gets the same codeword, possessed by the honest parties in CORE

  Restriction: allowed communication $\mathcal{O}(n^2 \log|\mathbb{F}|) + \mathcal{BC}(\mathcal{O}(n^2))$ bits

- Each P_i re-defines its component c_{ii} based on the communication it had received earlier from the parties only in CORE

Iff $v_{ij} = v_{ji} = 1$

$(c_{ii} = c_{ji})$ and $(c_{ij} = c_{jj})$ holds if P_i and P_j are honest

$c_i \overset{\text{def}}{=} \text{maj}(c_{1i}, \cdots, c_{(2t+1)i})$

- **Majority** of the parties in CORE will be honest
  - Redefined c_i will be the i^{th} component of the common codeword held by the honest parties in CORE

P_1
P_2
P_n
P_i
P_j
Sender
(c_{1i}, c_{11})
(c_{ji}, c_{jj})
$(c_{(2t+1)i}, c_{(2t+1)(2t+1)})$
(c_{ni}, c_{nn})
No Communication
c_{11} c_{12} ... c_{1n}
c_{j1} c_{j2} ... c_{jn}
c_{21} c_{22} ... c_{2n}
c_{n1} c_{n2} ... c_{nn}
P_{2t+1}
CORE
NPTEL, IISc

So, that is a conclusion till now; that means, we have; that means, we have ensured here we are ensured here that if at all a CORE is obtained then all the honest parties within the CORE have received the same codeword from the sender. So, without loss of generality, imagine CORE consists of the first $2t + 1$ parties, but it could be any subset of $2t + 1$ parties.

And it could be possible that there are some honest parties outside CORE whose local codeword might be different from the common codeword held by the honest parties in the CORE. So, that is why now to complete the protocol what is left is to ensure that every party P_i who is honest irrespective of whether it is a part of the CORE or not gets the same common codeword, which is guaranteed to be held by the honest parties in the CORE.

So, we have honest parties who may be inside the CORE as well as outside the CORE. Inside the CORE all the honest parties are guaranteed to be have the same code word, but as I said it could be possible that there are some honest parties outside CORE whose code words as received from the sender might be different from the common code words held by the honest parties inside the CORE.

So, to get rid of this mismatch we now have to ensure that even the honest parties outside the CORE gets the same common codeword. In general, our now goal is to ensure that every honest P_i irrespective of whether it is inside the CORE or outside the CORE gets the same common codeword as held by the honest parties in CORE. Now to do this there are several ways, but remember that we want a mechanism where the allowed budget for communication complexity is this much.

So, to respect this allowed budget what every party P_i will do is the following. Every party P_i will simply ignore whatever codeword or whatever the ith component of the codeword it has received from the sender because, it does not know whether sender was honest or whether sender was corrupt. Party P_i now knows that who are the parties in CORE because it also would have obtained the same code by finding the clique and what it knows is that, it is a parties in CORE who have the same common codeword specifically the honest parties.

So, what it will do is, it will try to redefine or it will try to adopt the ith component of its codeword to the ith component of the code words held by the honest parties in the CORE. Now ith component of the codeword held by the honest parties in CORE is guaranteed to be same because, all the honest parties in the CORE have the same common code word. So, how P_i redefines its ith component? Well, remember that earlier every party would have sent two common values to P_i as part of the pair wise consistency test ok all the n parties.

Now P_i will go back to that round and now it knows the identity of the parties in the CORE it will only focus on the ith component as received from the parties in CORE. At that time it would not know the size it would not know the identity of the core, but now it knows the identity of the CORE.

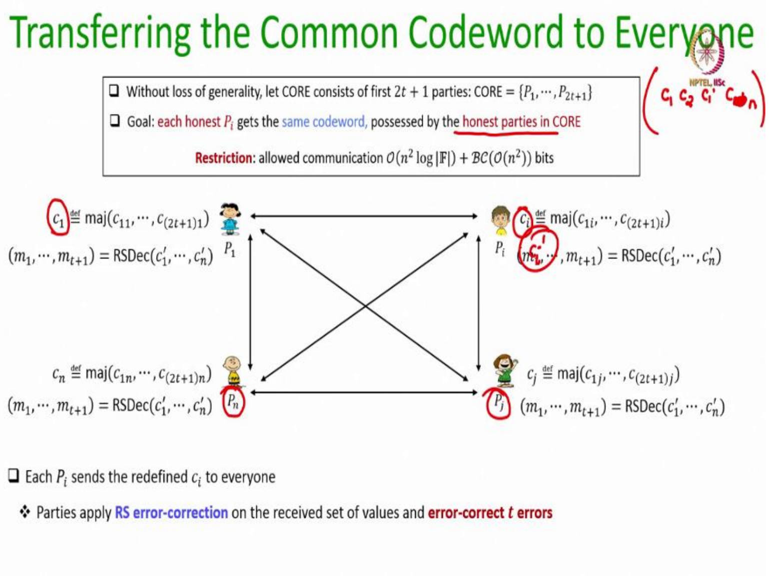
So, it simply ignores the ith component received from the other parties who are outside the CORE it only focuses on the ith component received from the parties in the CORE. And now whatever c_i it has obtained earlier from the sender it simply ignores it. So, remember sender would have sent a codeword to the party P_i and their ith component would have been there. Now P_i simply goes and forget that c_i and changes its c_i to this value.

Namely it takes the majority of the ith component of the code words of the parties in CORE as received earlier. Now what is the logic behind this redefining the ith component? Well we now know that the majority of the parties in CORE because the size of CORE is $2t + 1$.

So, $t + 1$ are honest and at most t could be corrupt and since all the honest parties in CORE would have sent the same ith component because, their respective code words are same. That will ensure that by taking by redefining the c_i based on this majority rule, the redefined c_i that P_i will have is same as the ith component of the common codeword held by the honest parties in CORE.

So, this step does not require any communication no communication; because this is based on communication which has happened earlier, but that time the identity of the CORE was not known, but now the identity of the CORE is known.

(Refer Slide Time: 42:06)



Transferring the Common Codeword to Everyone

- Without loss of generality, let CORE consists of first $2t+1$ parties: CORE $= \{P_1, \cdots, P_{2t+1}\}$
- Goal: each honest P_i gets the same codeword, possessed by the honest parties in CORE

Restriction: allowed communication $\mathcal{O}(n^2 \log|\mathbb{F}|) + \mathcal{BC}(\mathcal{O}(n^2))$ bits

$(c_1\ c_2\ c_i'\ c_n)$

$c_1 \overset{\text{def}}{=} \text{maj}(c_{11}, \cdots, c_{(2t+1)1})$ P_1
$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

P_i $c_i \overset{\text{def}}{=} \text{maj}(c_{1i}, \cdots, c_{(2t+1)i})$
$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$c_n \overset{\text{def}}{=} \text{maj}(c_{1n}, \cdots, c_{(2t+1)n})$ P_n
$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

P_j $c_j \overset{\text{def}}{=} \text{maj}(c_{1j}, \cdots, c_{(2t+1)j})$
$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

- Each P_i sends the redefined c_i to everyone
  - Parties apply RS error-correction on the received set of values and error-correct t errors

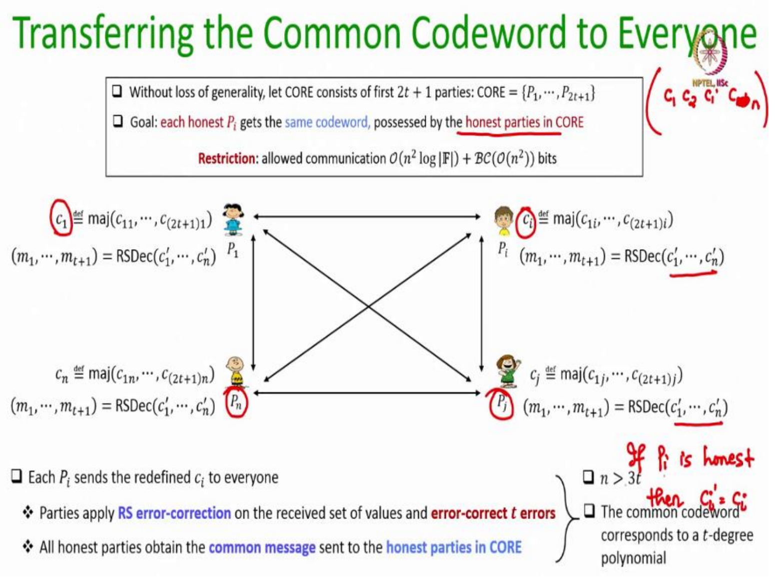
So, at this point each party will now have the common each party will now have its respective component of the common codeword held by the honest parties in the CORE ok. So, imagine that a common codeword held by the honest parties in the CORE has the component $c_1, c_2, \ldots, c_i, .., c_n$ because of the redefinition redefining step that we have executed just now.

P_1 will be having the component c_1, P_i will be having the component c_i, P_n will be having the component c_n and P_j will be having the component c_j ok, but that does not complete the protocol we now have to ensure that every party $P_1, P_2, \ldots, P_i, \ldots, P_n$ has this full vector namely the common codeword held by the honest parties in the CORE.

Because that exactly is the codeword which everyone would like to have which center has given only to the honest parties in the core. So, for this what we do is the following every party send its redefined component to everyone else. So, P_1 will send c_1 to everyone, P_i will send c_i to everyone, P_j will send c_j to everyone and P_n will send c_n to everyone.

This will require a communication of n^2 field element of course, if P_i is corrupt then it may change c_i to c_i' or it may send a garbage c_i to parties. So, what the parties can now do is the following. So, let us denote the vector of values that each party now have after this exchange as $c_1', c_2', \ldots, c_n'$.

(Refer Slide Time: 43:59)



Transferring the Common Codeword to Everyone

- ❑ Without loss of generality, let CORE consists of first $2t+1$ parties: CORE = $\{P_1, \cdots, P_{2t+1}\}$
- ❑ Goal: each honest P_i gets the same codeword, possessed by the honest parties in CORE

Restriction: allowed communication $\mathcal{O}(n^2 \log|\mathbb{F}|) + \mathcal{BC}(\mathcal{O}(n^2))$ bits

$(c_1\, c_2\, c_i'\, c_n)$

$c_1 \overset{\text{def}}{=} \text{maj}(c_{11}, \cdots, c_{(2t+1)1})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$c_i \overset{\text{def}}{=} \text{maj}(c_{1i}, \cdots, c_{(2t+1)i})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$c_n \overset{\text{def}}{=} \text{maj}(c_{1n}, \cdots, c_{(2t+1)n})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$c_j \overset{\text{def}}{=} \text{maj}(c_{1j}, \cdots, c_{(2t+1)j})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

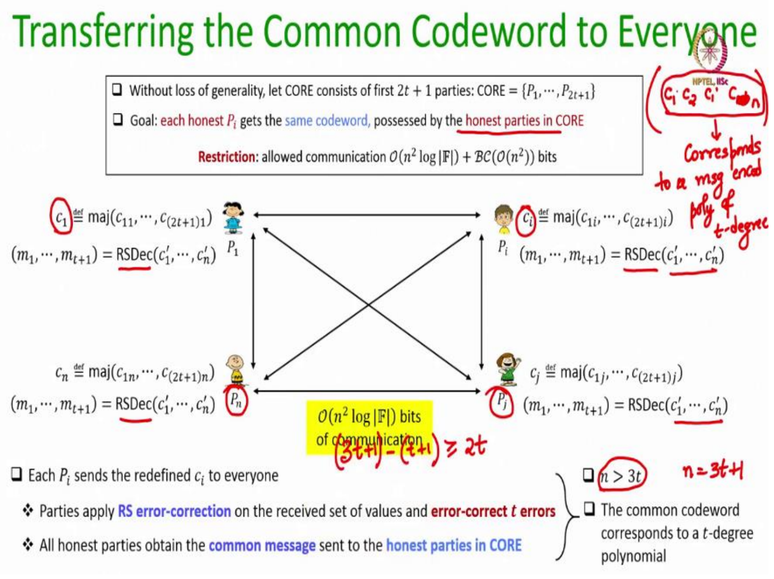
- ❑ Each P_i sends the redefined c_i to everyone
  - ❖ Parties apply **RS error-correction** on the received set of values and **error-correct t errors**
  - ❖ All honest parties obtain the **common message** sent to the **honest parties in CORE**

- ❑ $n > 3t$
- ❑ The common codeword corresponds to a t-degree polynomial

If P_i is honest then $c_i' = c_i$

If P_i is honest then c_i' will be guaranteed to be same as c_i, but if P_i is corrupt then c_i' could be a garbage value, but what we now know is that we are working in the setting where $n > 3t$.

(Refer Slide Time: 44:28)



Transferring the Common Codeword to Everyone

- Without loss of generality, let CORE consists of first $2t+1$ parties: CORE = $\{P_1, \cdots, P_{2t+1}\}$
- Goal: each honest P_i gets the same codeword, possessed by the honest parties in CORE

Restriction: allowed communication $\mathcal{O}(n^2 \log|\mathbb{F}|) + \mathcal{BC}(\mathcal{O}(n^2))$ bits

$c_1 \overset{\text{def}}{=} \text{maj}(c_{11}, \cdots, c_{(2t+1)1})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$c_i \overset{\text{def}}{=} \text{maj}(c_{1i}, \cdots, c_{(2t+1)i})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$c_n \overset{\text{def}}{=} \text{maj}(c_{1n}, \cdots, c_{(2t+1)n})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$c_j \overset{\text{def}}{=} \text{maj}(c_{1j}, \cdots, c_{(2t+1)j})$

$(m_1, \cdots, m_{t+1}) = \text{RSDec}(c_1', \cdots, c_n')$

$\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits of communication

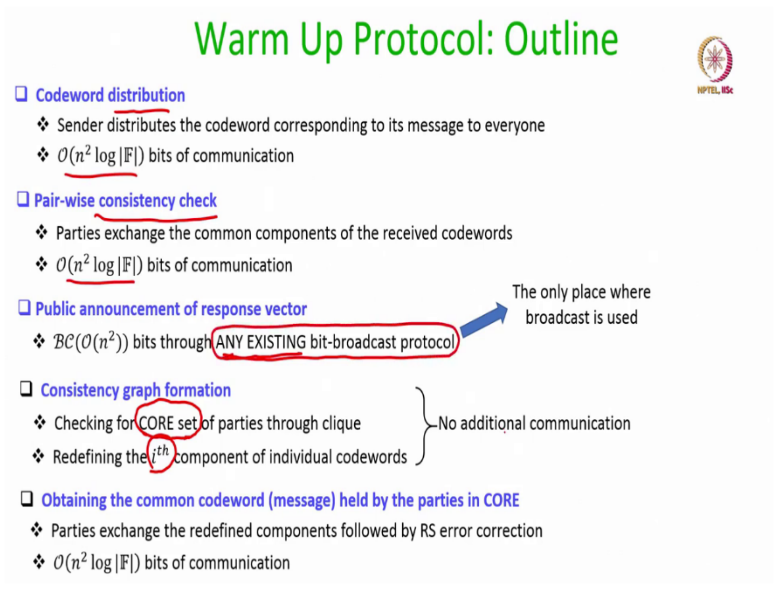
- Each P_i sends the redefined c_i to everyone
  - Parties apply **RS error-correction** on the received set of values and **error-correct t errors**
  - All honest parties obtain the **common message** sent to the **honest parties in CORE**

- $n > 3t$
- The common codeword corresponds to a t-degree polynomial

And $c_1 c_2, \ldots, c_n$ corresponds to a message encoding polynomial of t degree ok. Specifically, we are working in the case where $n = 3t + 1$. So, $3t + 1$ is the length of the codeword and what is the length of the message encoding polynomial? It has $t + 1$ coefficient's and how many parties can send incorrect redefined components, up to t. So, this condition is satisfied; that means, whatever condition needs to be satisfied for Reed Solomon error correction for error correcting t errors will be satisfied.

And as a result, every party after applying the Reed Solomon decoding algorithm on the received vector of redefined components will be able to recover back the message corresponding to the common codeword held by the honest parties in core and this will require a communication of only n^2 bits, n^2 field elements.

(Refer Slide Time: 45:41)



Warm Up Protocol: Outline

- Codeword distribution
  - Sender distributes the codeword corresponding to its message to everyone
  - $\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits of communication
- Pair-wise consistency check
  - Parties exchange the common components of the received codewords
  - $\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits of communication
- Public announcement of response vector
  - $\mathcal{BC}(\mathcal{O}(n^2))$ bits through ANY EXISTING bit-broadcast protocol

The only place where broadcast is used

- Consistency graph formation
  - Checking for CORE set of parties through clique
  - Redefining the i^{th} component of individual codewords

No additional communication

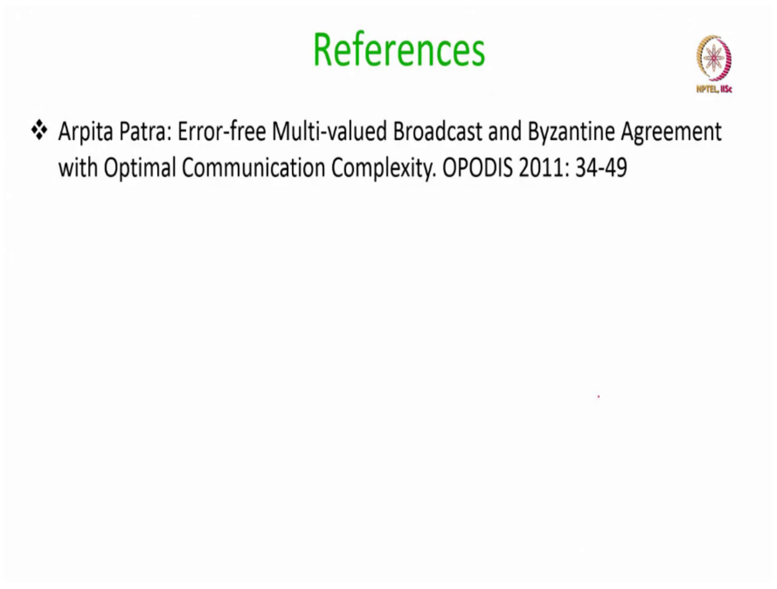
- Obtaining the common codeword (message) held by the parties in CORE
  - Parties exchange the redefined components followed by RS error correction
  - $\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits of communication

So, if we outline this warm up protocol it has multiple stages the first stage is the code distribution where sender converts it message into a code word and sends the codeword to everyone. And then the second stage is the pair wise consistency test where every pair of parties exchange only a constant number of supposedly common points and then they publicly prepare and then they publicly announce the response vector. For this step any existing bit broadcast protocol can be used based on the publicly available response vectors the parties prepare a consistency graph.

And then they check whether there exist a sufficiently large subset of parties a CORE subset of parties of size $2t + 1$, who are all pair wise consistent guaranteeing that at least $t + 1$ honest parties have received a common codeword if that is the case then, based on the communication the parties had in this pair wise consistency check phase from the parties in CORE every party redefines the ith component of their individual code words.

This does not require any communication and once every party has their respected redefined component, the goal will be to obtain the common codeword held by the parties in the code for this every party exchange the redefined component followed by Reed Solomon error correction this will require a communication of n^2 field elements. So, you can see most of the protocol only involves communication over point to point channels the existing bit broadcast protocol instances is required only to make the response vector public.

(Refer Slide Time: 47:32)



References

❖ Arpita Patra: Error-free Multi-valued Broadcast and Byzantine Agreement with Optimal Communication Complexity. OPODIS 2011: 34-49

So, this is the reference used for the warm up protocol. In the next lecture we will see how we can plug in this warm up protocol to get the actual domain extension protocol.

Thank you.