

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


Lecture - 26
Domain Extension for Perfectly-Secure Broadcast Based on RS Error-Correcting Codes: I

Hello, everyone. Welcome to this lecture. So, for the next few lectures we will be now focusing on Domain Extension for Perfectly – Secure Broadcast. Earlier, we had seen domain extension for byzantine agreement namely, the domain extension by Turpin and Coan, and at that time I promised that later on we will see more efficient domain extension protocol.

So, we will now see the more efficient domain extension based on the properties of Reed – Solomon error correcting codes.

(Refer Slide Time: 00:55)

Lecture Overview



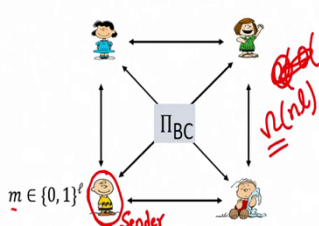
- ❑ Domain extension for broadcast
 - ❖ Problem description
 - ❖ Some properties of RS error-correcting codes

So, we will start with the problem description and before going into the exact protocol for domain extension, we will see some additional properties of Reed-Solomon error correcting codes which will be useful, ok.

(Refer Slide Time: 01:10)

From Bit-broadcast to Broadcast for Any Domain

□ We have seen various bit-broadcast protocols with $n > 3t$



$m \in \{0, 1\}^l$ **Sender**

□ What if the input of the sender is from $\{0, 1\}^l$?

- ❖ **Option I:** Run any bit-broadcast protocol ℓ times $\rightarrow \ell \cdot BC(1)$
- ❖ **Option II: Domain extension,** run any bit-broadcast protocol a constant number of times $\rightarrow O(n\ell)$

➤ Turpin-Coan domain extension: $O(n^2\ell) + BC(1)$
 ➤ We will now see a **more efficient** domain extension with $O(n\ell) + BC(1)$ communication cost

Handwritten notes:
 asymptotically optimal
 For sufficiently large ℓ , $O(n^{t+1})$
 the total cost = $O(n\ell)$ bits
 ❖ EIG protocol with $O(n^{t+1})$ bits of communication
 ❖ Phase-king protocol with $O(n^3)$ bits of communication
 denotes the communication complexity for 1 instance of bit-broadcast if $|L| = \Theta(n^2)$
 much better than Turpin-Coan's
 $O(n\ell) + O(n^2)$
 $O(n\ell) + BC(1)$

So, what exactly is the goal of domain extension for broadcast and byzantine agreement? So, we will be focusing on the domain extension for broadcast, since the broadcast and byzantine agreement problems are equivalent to each other in the honest majority setting, in the sense that a protocol for one problem implies protocol for other.

So, you can imagine that if we have a if we have an efficient domain extension protocol for broadcast then we can use it to get a domain extension protocol for byzantine agreement as well. So, in the context of broadcast the goal for domain extension is as follows. We have seen various bit broadcast protocols with $n > 3t$, ok. Say for instance the EIG protocol, the phase-king protocol with the various communication complexities.

And, these are bit broadcast protocols where the sender's message is a single bit. Now, what if the input of the sender is from a bigger set? Say the sender input is no longer is just a single bit, but rather it is a bit string of size ℓ bits. So, there are two ways to achieve broadcast for ℓ bits – one option is that we run any bit broadcast protocol ℓ times where the input for the i -th instance of the bit broadcast will be the i -th bit of the sender, ok.

And, then the overall output for the parties will be the concatenation of the individual outputs, which the parties receive from the various instances of the bit broadcast namely from the ℓ instances of the bit broadcast. However, this will require a communication complexity which will be ℓ times the communication cost of one instance of bit broadcast.

So, this notation BC_1 and within the parenthesis 1 it denotes the communication complexity for one instance of bit broadcast. Say for instance, if we are considering the EIG protocol then BC_1 for EIG protocol is $n^t + 1$; that means, if we run l instances of the EIG protocol the overall cost will be order of $n^t + 1$ times l whereas, if we use the phase king protocol as the bit broadcast protocol, then the overall complexity will be order of n^3 times l ok.

The domain extension which is the option number 2 is basically to design a broadcast protocol where the senders input is now of size l bits where the existing bit broadcast protocols are executed only a constant number of times independent of the size of l , ok and we had already seen earlier the Turpin Coan domain extension protocol there the communication complexity of the overall broadcast protocol was this. Of course, we had seen the Turpin Coan domain extension in the context of byzantine agreement, but as I said the same extension will work even for the broadcast with the same complexity.

So, now, you can see that from the communication complexity expression the cost which depends upon l , it is only over the point to point channel namely this order of $n^2 l$ bits of communication happens over the point to point channels and existing bit broadcast protocols are executed only for a constant number of times which is independent of l .

Now, we will now see a more efficient domain extension protocol where the overall communication complexity will turn out to be this much ok. So, now, you can see that this is much better than Turpin Coans protocol and this communication complexity of order of nl plus the cost of bit broadcast will be asymptotically optimal communication wise it will be asymptotically optimal.

This is because if we substitute the cost of the bit broadcast say by the phase king bit broadcast protocol then the overall cost turns out to be order of $n^3 l$ plus order of n^2 ; that means, if the message size small l is guaranteed to be say $\theta(n^2)$; that means, if you have a sufficiently large message then this expression order of nl plus order of n^3 will become order of nl namely, the term order of n^3 which is coming due to this constant number of invocations of the bit broadcast will be asymptotically subsumed in this order notation big O of nl .

That means asymptotically I can claim that for sufficiently large l for sufficiently large l the total cost will turn out to be order of $n^2 l$ bits because whatever is the additional term

coming due to the bit broadcast instances will be subsumed asymptotically in this order of $n \log n$ expression. Now, order of $n \log n$ bits of communication is obviously, an optimal communication complexity for any broadcast protocol where the sender's message is of size $\log n$ bits.

Because since it is a broadcast it needs to be guaranteed that the sender's message is communicated to all the parties at least once. Of course, in the protocol the parties may exchange messages further, but in order that every party obtains the sender message. The sender has to send its message to every other party and that trivially requires a communication complexity of order of $n \log n$, sorry $\Omega(n \log n)$, that is a minimum communication required in any broadcast protocol where the sender's message is of size $\log n$ bits.

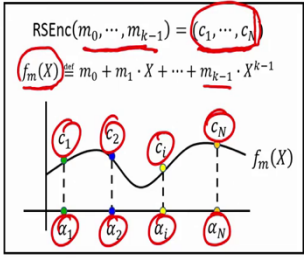
And, what is the total communication complexity we will be achieving through the more efficient domain extension bit broadcast protocol? That will be order of $n \log n$; that means, we will be asymptotically achieving the minimum cost which is expected from any broadcast protocol for message size of $\log n$ bits, ok.

(Refer Slide Time: 09:21)

RS Error-Correcting Codes: More Properties

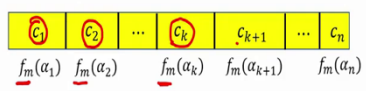
$m \in F^k$

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

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


Reed-Solomon Encoding

$\alpha_1, \alpha_2, \dots, \alpha_i, \alpha_n$



- Any k components of the codeword vector, uniquely determine the complete codeword vector
 - ❖ k components correspond to k distinct points on $f_m(X)$. $\therefore f_m(X)$ is a $(k-1)$ -degree polynomial
 - ❖ Any k distinct points on $f_m(X)$ uniquely determine $f_m(X)$ and hence the remaining points on $f_m(X)$

So, before going into our exact domain extension protocol, we need to understand some more properties of Reed – Solomon error correcting codes.

So, just to recap this is the Reed – Solomon encoding algorithm where if the sender has a message m consisting of k elements m_0, \dots, m_{k-1} from the field F , then to compute the Reed – Solomon code word a message encoding polynomial

$f_m(X) = m_0 + m_1X + \dots + m_{k-1}X^{k-1}$ is formed which is a $k - 1$ degree polynomial. And then that polynomial is evaluated at n publicly known values $\alpha_1, \alpha_2, \alpha_i, \alpha_N$ and they constitute the individual components of the code word.

That is a Reed – Solomon encoding algorithm. Now, the first property that we require later in our domain extension protocol is the following.

Any k components of this code word vector, uniquely determine the remaining components of the code word vector; that means, you do not need the full code word vector to identify what exactly was the senders message. Even if I give you any k components of this code word vector and those k components need not be the consecutive k components, they could be any subset of the k components of this full vector. That will be sufficient to determine the remaining components of the code word.

So, for demonstration imagine that I give you say the first k components, but as I said whatever I am claiming here holds for any k component. So, imagine you are given only the k components of the code word and you do not you are not given the remaining components, they are unknown for you. Of course, you know the value of $\alpha_1, \alpha_2, \alpha_i, \alpha_n$ because they are publicly known.

And, you do not know the value of the underlying polynomial underlying message encoding polynomial, but what you know is that the components, which are given to you they are basically the value of the so called message encoding polynomial at the corresponding α points. That is what is known to you. Now, the k components which are given to you basically constitute k distinct points on the polynomial $f_m(X)$ namely the message encoding polynomial.

And, what was the degree of the message encoding polynomial? Well, it is a k minus 1 degree polynomial that information is available to you what is a degree because that is a system parameter. Now, you are given k distinct points on a k minus 1 degree polynomial and recall that we know from Lagrange's interpolation that any k distinct points on an unknown k minus 1 degree polynomial are sufficient to uniquely determine that unknown polynomial.

So, through the components which are given to you the k components you can interpolate the message encoding polynomial and once the message encoding polynomial is computed, well

you can evaluate it at the remaining alpha values to find out what were the missing components, that is all, right.

So, we will be extensively using this property. That means it is not necessary to know this full vector of size n to determine the message encoding polynomial or this message, which was the input for the Reed – Solomon encoding algorithm. Knowledge of any k components is sufficient.

(Refer Slide Time: 13:27)

RS Error-Correcting Codes: More Properties

$m = (m_0, m_1, \dots, m_{k-1})$

$c_{1,1} = f_1(\alpha_1)$
 \vdots
 $c_{1,k} = f_1(\alpha_k)$
 \vdots
 $c_{1,N} = f_1(\alpha_N)$

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

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

$m' = (m'_0, m'_1, \dots, m'_{k-1})$

$c_{2,1} = f_2(\alpha_1)$
 \vdots
 $c_{2,k} = f_2(\alpha_k)$
 \vdots
 $c_{2,N} = f_2(\alpha_N)$

$f_1(X) \equiv m_{1,0} + m_{1,1} \cdot X + \dots + m_{1,k-1} \cdot X^{k-1}$

 $c_{1,1} = c_{2,1}$
 \vdots
 $c_{1,k} = c_{2,k}$
 $f_2(X) \equiv m_{2,0} + m_{2,1} \cdot X + \dots + m_{2,k-1} \cdot X^{k-1}$

$\Rightarrow f_1(x) \text{ and } f_2(x) \text{ have at least } k \text{ common values}$

$\Rightarrow f_1(x) = f_2(x)$
 $\Rightarrow m' = m$

❑ If two RS codewords have the same k or a greater number of components, then both the codewords are the same and correspond to the same message

❖ Two different $(k-1)$ -degree polynomials can have at most $(k-1)$ common values

Now, let us see some more properties. The next property that we will be using later is the following. If you are given two Reed – Solomon code words corresponding to two messages and if it so happens that two Reed – Solomon code words have the same k or more number of components ok. Then both the code words correspond to the same message and they are actually the same code words they are not the different code words.


So, what I am saying here is the following. So, imagine you are given two codewords $(c_{11}, c_{12}, \dots, c_{1n})$ and $(c_{21}, c_{22}, \dots, c_{2n})$ which are component-wise same at k or more locations. Now, imagine that the first code word corresponds to a message m with message encoding polynomial $f_m(X)$. And say that the second code word corresponds to a message m' with message encoding polynomial $f_{m'}(X)$.

Now, imagine that these two code words have the same k or greater number of components, say for simplicity these both these two code words have the first k components being the same, ok. Then it implies that $f_m(X) = f_{m'}(X)$. This is because the common components of the two codewords constitute k distinct points on the two polynomials. And two different $k - 1$ degree polynomials can have at most $k - 1$ common points.

So, that is the second property of the Reed – Solomon error correcting codes which will be useful later when we discuss the exact domain extension protocol namely. Two different RS code words corresponding to messages of size k can have at most k minus 1 common component common components. They cannot have more they cannot have k or more number of common components.

(Refer Slide Time: 19:15)

References



- ❖ Matthias Fitzi, Martin Hirt: Optimally efficient multi-valued byzantine agreement. PODC 2006: 163-168 ✓
- ❖ Arpita Patra: Error-free Multi-valued Broadcast and Byzantine Agreement with Optimal Communication Complexity. OPODIS 2011: 34-49 ✓

So, these are the references which are used for today's lecture. The domain extension based on the properties of Reed – Solomon error correcting codes were first studied in this paper and then a more efficient version of the protocol was proposed in this follow up work.

Thank you.