

**Secure Computation - Part I**  
**Prof. Ashish Choudhury**  
**Department of Computer Science**  
**International Institute of Information Technology, Bangalore**

**Module - 1**  
**Lecture - 7**  
**Secret Sharing**

**(Refer Slide Time: 00:32)**

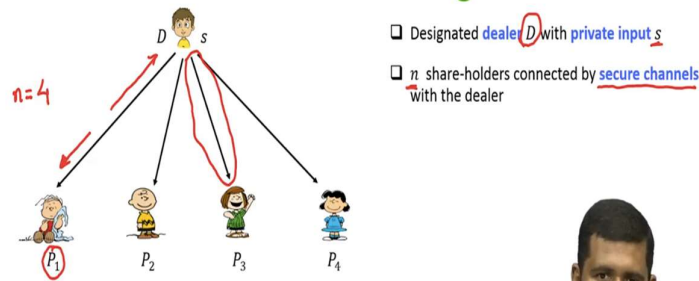
## Lecture Overview

- Secret Sharing
  - ❖ Problem definition
  - ❖ Threshold secret-sharing scheme

Hello everyone. Welcome to this lecture. So, the plan for this lecture is as follows: We will start with the problem of secret-sharing, which is one of the fundamental primitives which we are going to use in our MPC protocols. And as part of secret-sharing, we will first try to understand the problem definition of secret-sharing and we will see a special case of secret-sharing, namely threshold secret-sharing.

**(Refer Slide Time: 01:02)**

## Secret-Sharing



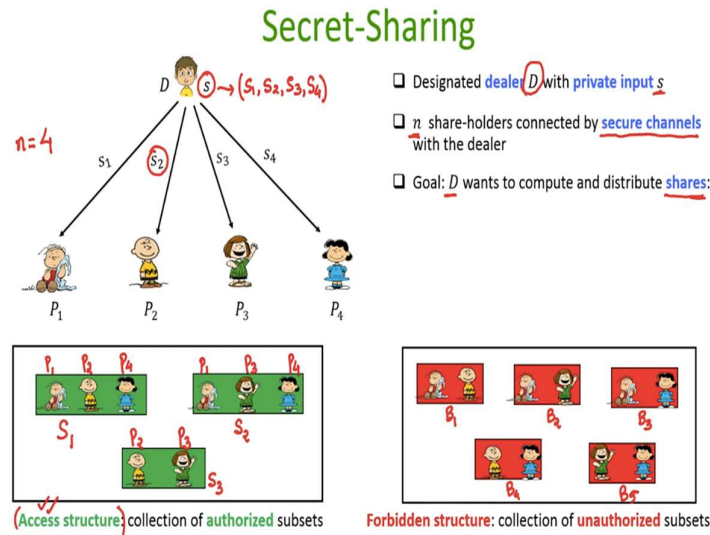
So, let us start with the problem of secret-sharing. So, the setting is as follows: We have a designated party which we often call us dealer, and denoted by  $D$ . And it has some private input, which I denote by  $s$ . So, it is an abstract input. It could be a single bit, it could be your file, it could be any kind of data which is known only to the dealer; or it could be, for example, your AES encryption key and so on.

Now, there are  $n$  parties denoted by  $P_1, P_2, P_3, P_4, P_n$ . So, for instance, in this particular example, I am taking the case where we have 4 shareholders, namely where  $n = 4$ , but in general, you can have up to  $n$  number of shareholders. And each of these shareholder is connected with the dealer by what we call as secure channel. That means, if I consider for instance party number  $P_1$  as part of the setting, it is given to us that there is a mechanism by which dealer can communicate privately to this party  $P_1$ .

And by privately I mean that, whatever communication happens over this channel between dealer and  $P_1$ , no other party can find out what exactly is the communication that is happening between  $D$  and the party  $P_1$  over this channel. In the same way, if I consider, say for instance, the channel between  $D$  and party number  $P_3$ , only  $D$  and  $P_3$  can communicate over this channel and no other party can make out anything regarding the communication which is happening over this channel between  $D$  and  $P_3$ .

So, that is what we mean by secure channel. So, as part of the setting, it is given to us that there is a mechanism by which dealer can communicate privately and securely to, with any of these  $n$  shareholders.

(Refer Slide Time: 03:32)



We are also given 2 different collections of parties or these shareholders. One collection is what we call us access structure, which can be interpreted as some kind of a collection of authorised subsets. So, for instance, I can have one authorised subset consisting of  $P_1, P_2, P_4$ ; or I can have an authorised subset consisting of party  $P_1$ , party  $P_3$ , party  $P_4$ ; or my authorised subset could be party  $P_2$  and party  $P_3$ .

So, collection of this authorised subsets which is also called as access structure will be given to you as part of the problem definition. And with respect to this access structure, we have a complimentary structure which we call as forbidden structure, which is a collection of unauthorised subsets. So, the authorised subsets, they are denoted in green colour; the unauthorised subsets, they are in red colour.

So, given the authorised subsets, I can compute the unauthorised subsets. I will very soon discuss how exactly we compute the unauthorised subsets, given the authorised subsets. So, with respect to this specific example, if I am given subset number 1, subset number 2 and subset number 3, with respect to these 3 possible authorised subsets, I have 5 forbidden structures or unauthorised structures, subsets, which I call as, denote as  $B_1, B_2, B_3, B_4$  and  $B_5$ .

So, what are the things given to you? You are given the fact that there is a designated dealer and it has a private input. The value of the private input is not known, but it will be known that its input from some specific space. And you are given a collection of authorised subsets. And

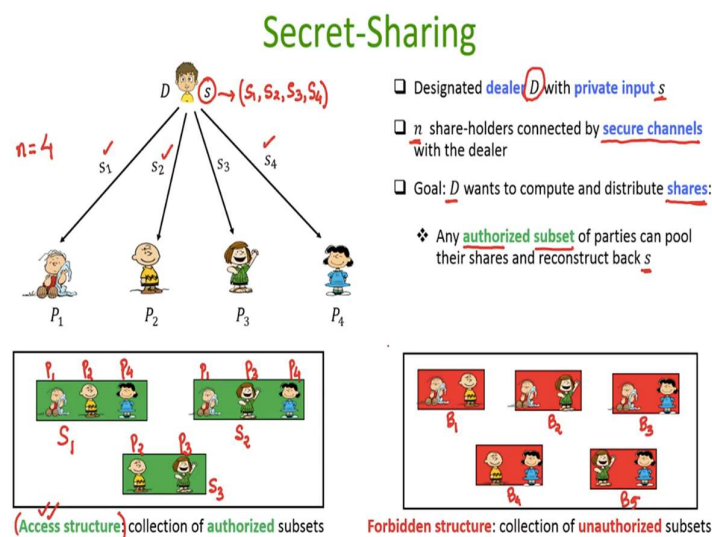
with respect to those authorised subsets, you have another collection consisting of unauthorised subsets. Now, what is the goal?

We want a mechanism which allows the dealer to compute what we call as shares, shares for his secret  $s$ . So, since there are  $n$  shareholders, we want a mechanism which allows the dealer to compute  $n$  shares and distribute the shares to the respective shareholders over the corresponding private channel. So, for instance, here we have 4 shareholders. So, we want a mechanism which allows the dealer to compute 4 shares,  $s_1, s_2, s_3$  and  $s_4$ , out of its secret  $s$ .

And the shares are computed and the shares are distributed to the respective shareholders. So, the first share which I denote as  $s_1$ , dealer will communicate over the private channel to  $P_1$ . And when  $s_1$  is communicated to  $P_1$ , none of the remaining 3 parties can make out what is the value of  $s_1$ , which dealer has given to  $P_1$ . The second share  $s_2$ , dealer will give over the private channel to party number  $P_2$ , and none of the remaining 3 parties  $P_1, P_3, P_4$ , can make out what is the value of  $s_2$ .

In the same way, the share  $s_3$  is given to party  $P_3$ , and share  $s_4$  is given to party  $P_4$ . If there would have been  $n$  shareholders, there dealer would have computed  $n$  shares, and each share would have been communicated to the corresponding shareholder. Now, we want 2 properties to be achieved by this sharing mechanism.

**(Refer Slide Time: 08:07)**

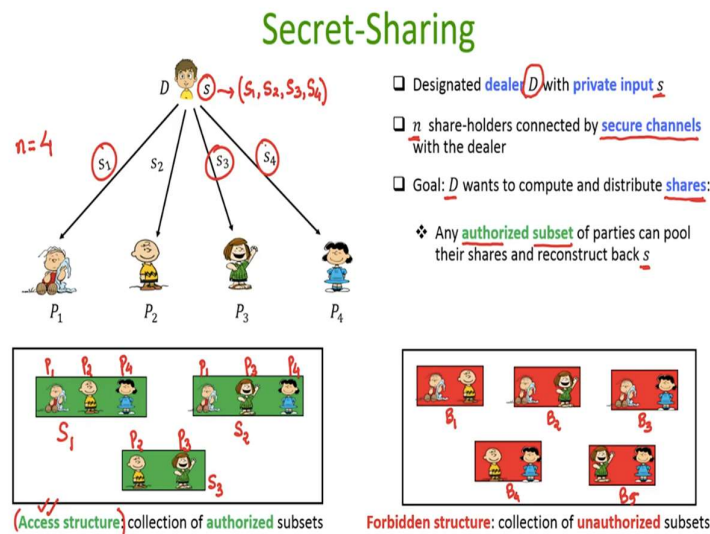


The first property is the following: If we consider any authorised subset from your access structure, it could be either the collection number  $S_1$  or the collection number  $S_2$  or the

collection  $S_3$ . You take any authorised subset of shareholders, there should be a mechanism for them to kind of combine their shares and get back the secret  $s$ . And that is why they are called as authorised subset.

They are authorised in the sense that they are allowed, they are supposed to get back the secret  $s$  if they somehow combine their shares; there should be a mechanism to get back the secret  $s$  from their shares. So, for instance, what I am saying is, they should; the sharing mechanism or computing the shares from the secret, that process should be such that, if we take the share number  $s_1$ , share number  $s_2$  and share number  $s_4$ , then, there should be a mechanism to get back the secret  $s$ . So, because  $P_1, P_2, P_4$  constitutes an authorised subset.

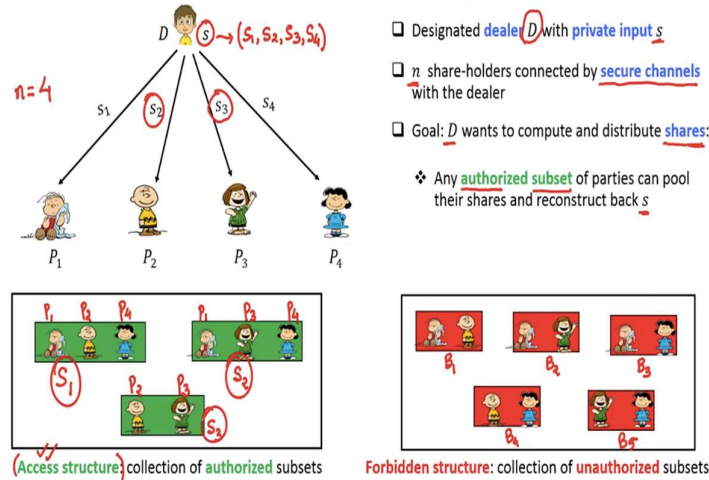
**(Refer Slide Time: 09:21)**



In the same way, there should be a mechanism to combine your share number  $s_1$ , share number  $s_3$  and share number  $s_4$  to get back the secret.

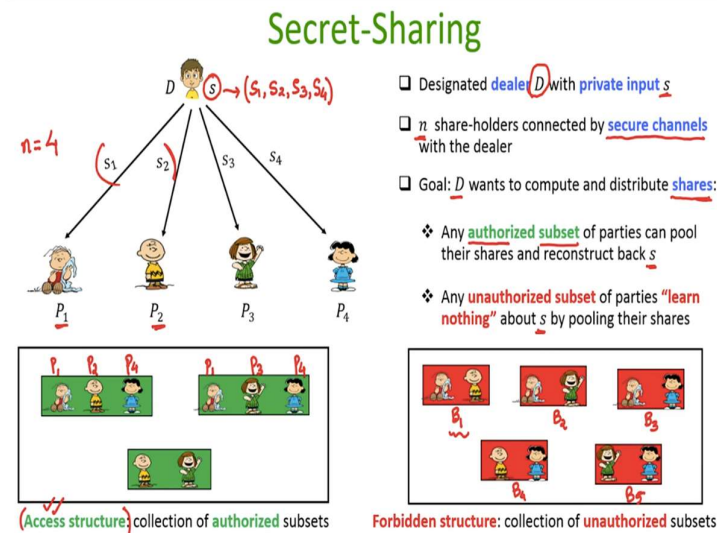
**(Refer Slide Time: 09:35)**

## Secret-Sharing



And there should be a mechanism to combine your share  $s_2$  and share  $s_3$ , and get back your secret  $s$ . Because,  $S_1, S_2, S_3$ , the subset  $S_1$ , subset  $S_2$ , subset  $S_3$ , they constitute authorised subsets. So, that is the first requirement from this sharing mechanism.

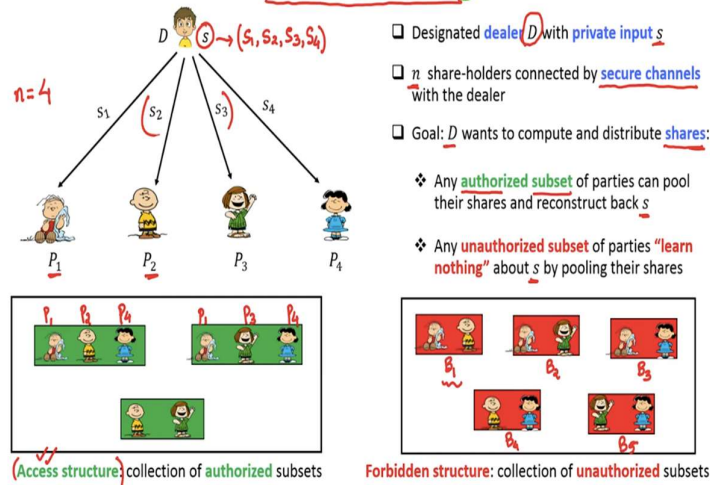
(Refer Slide Time: 09:57)



And the second property that we need here is the following: If any unauthorised subset of parties combine their shares, then they should not learn anything about the secret  $s$ . That means, what I am saying here is the following: If I consider, say the unauthorised subset  $B_1$ , which is consisting of  $P_1$  and  $P_2$ ; that means, just by combining the shares  $s_1$  and  $s_2$ , the secret should not be reconstructed back.

(Refer Slide Time: 10:40)

# Secret-Sharing

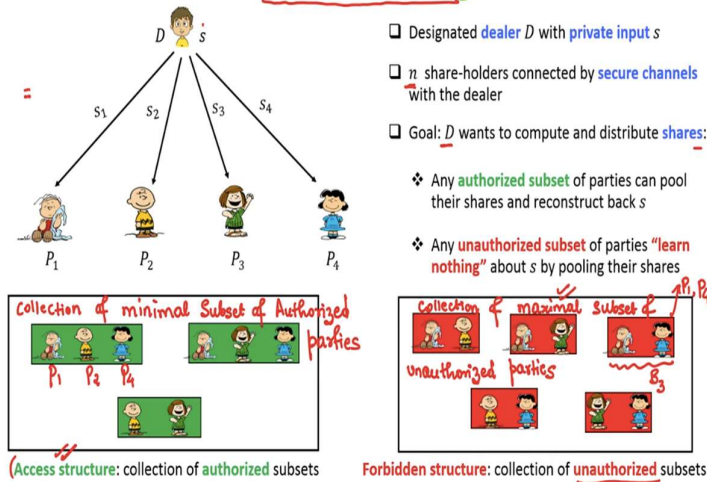


In the same way, if I consider the shares  $s_2$  and  $s_3$ , then there should not be any mechanism to get back the secret  $s$  by combining these 2 shares and so on. So, that is a problem of secret-sharing. And now, you can imagine that why this problem is called as secret-sharing. You want to share your secret, you want to compute shares of the secret and distribute to individual shareholders, so that certain specific collection of parties should be able to get back the secret by combining their shares.

Whereas, there are some blacklisted or unauthorised subset of parties who should not be able to get back the secret by combining their shares. Now, let us try to understand that how exactly we get this collection of unauthorised subsets from given the collection of authorised subsets. So, I will demonstrate it with this specific example only, but you can generalise it for the case when there are  $n$  parties.

**(Refer Slide Time: 11:42)**

# Secret-Sharing



So, the first thing to understand here is the following: That this access structure is basically the collection of what we call as minimal subset of authorised parties. So, what do we mean by minimal subset of authorised party? Minimal in the sense that, if, say for instance  $P_1$ ,  $P_2$  and  $P_4$ , together they are eligible to get back the secret  $s$ . Then, if we include any additional party in this collection, then together, they also are eligible to get back the secret.

So, for instance, in this collection  $P_1, P_2, P_4$ , if I also include  $P_3$ , by default, that whole collection, the bigger collection, that is also authorised to get back the secret  $s$ . That means, any superset of any authorised subset is by default is also an authorised subset. So, that is why, when we are specifying the access structure, we specify the collection of only minimal subsets, the minimal collection of parties which you require to get back the secret  $s$ .

That also means that you take any superset of them, they are also authorised. So, we have this minimal collection or minimal subsets of authorised parties. Now, the unauthorised parties are what we call as collection of maximal subset of unauthorised parties. So, there is a difference here. For the case of access structure, that was the collection of minimal subsets of authorised parties.

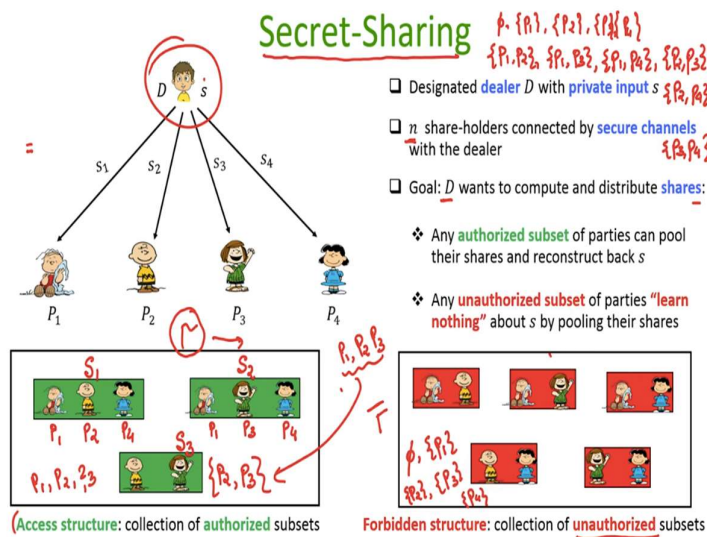
But when we are taking or talking about forbidden structure, that is a collection of maximal subsets, not minimal subsets. It is a collection of maximal subsets in the sense that, if I consider this specific forbidden set  $B_3$  which consists of party number  $P_1$  and  $P_4$ , it means that, even  $P_1$  and  $P_4$ , if they come together and combine their shares, they should not be able to learn the secret  $s$ .



That also means that, even if  $P_1$  alone tries to learn the secret, he should fail. Or, if  $P_4$  alone tries to learn the secret, she should fail. That means, any proper subset of  $B_3$  will also be considered as an unauthorised subset. This is just a reverse of what was the case for the authorised subset. For the authorised subset, any superset of authorised subset is also authorised.

For the case of unauthorised subsets, any proper subset of it will be also unauthorised. So, that is why we focus on the largest possible or maximal possible subsets of unauthorised sets. And that collection is called as the forbidden structure. So, now, let us see how we computed this forbidden structure. So, we consider what we call as the power subset of the shareholders. So, in this case, we have 4 parties.

(Refer Slide Time: 15:47)



And the power subset will consist of all possible subsets of these 4 parties. So, you have the null set; and then you have the singleton sets consisting of just 1 party each; and then you have collections consisting of 2 parties; and then you can write down the subsets consisting of 3 parties; and then the whole set. Now, what we are going to do is the following: You are given the authorised subsets.

So, you are given, say the subset number  $S_1, S_2$  and  $S_3$ . Basically, what we are trying to do here is, we are trying to take the compliment of the power set; we are trying to compute the compliment of the access structure with respect to the power set and focusing only on the maximal subsets. Roughly, that is our forbidden structure. So, what does that mean? So, this is; so, I have not written down the full power set; you have other sets as well.

So, by default  $\phi$  is always going to be a member of forbidden structure. Because, if no party comes together, then you are not supposed to get back the secret. So,  $\phi$  is definitely a member of your forbidden structure, because that will be present in the complement of your access structures. If my access structure is  $\Gamma$ , then basically, what I have constructed here is what I call as the complement of  $\Gamma$ ; but I would not be writing down all the sets in my complement of  $\Gamma$ , but I will be focusing on the maximal subsets.

So, of course,  $\phi$  is a member of gamma compliment, but it would not be represented in my forbidden structure, because that is implicitly there, that is implicitly present. In the same way,  $P_1$  will be present in gamma complement,  $P_2$  is also present in the gamma complement; singleton sets, all the singleton sets. They are going to be present in the forbidden structure, but I would not be writing down explicitly because they are implicitly present.

Because, as I said, any proper subset of the maximal subsets in your forbidden structure are also unauthorised; so, they are also going to be present in gamma complement. Now, I take the subsets consisting of 2 parties. So, it turns out that except the collection  $P_2, P_3$  which is an authorised subset, all other remaining subsets consisting of 2 parties will be present in forbidden structure, and that I have listed down.

Now, I take the collection of subsets of size 3. So, there are 2 subsets of size 3 which are authorised. So, those subsets are  $P_1, P_2, P_4$ ; and you have  $P_1, P_3, P_4$ . So, what about the case  $P_1, P_2, P_3$ ? So,  $P_1, P_2, P_3$ , even though it is not explicitly present in the access structure, it is present because that is a superset of this subset  $P_2, P_3$ . So, that is why  $P_1, P_2, P_3$  is also present here.

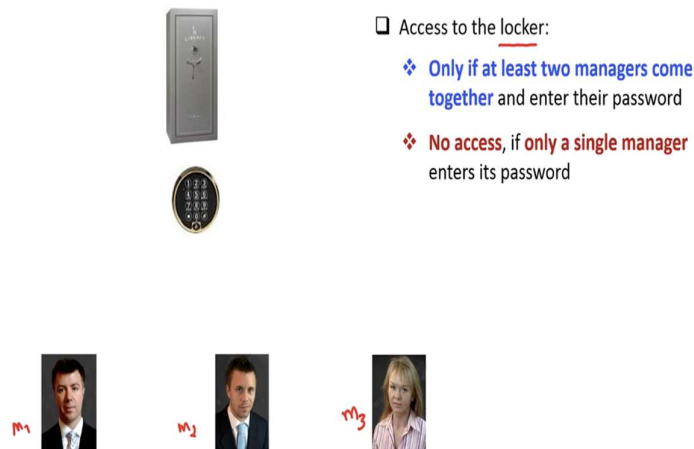
Because, along with 2 and 3, you include any party, that also be considered as an access structure. So, that is why, I am not writing down  $P_1, P_2, P_3$  in the complement of gamma, because that is present in gamma actually, implicitly. So, basically, what you have to do here is, you have to expand your gamma with respect to the minimal subsets; include all proper supersets as well in gamma.

That will be your actual, bigger authorised subsets access structure. And now you take the complement of that set with respect to your universal set which is the power set of your

shareholders, that will give you the forbidden structure. So, that is how you compute your forbidden structure with respect to your access structure. So, coming back to the problem of secret-sharing, basically, our goal is that we want to compute or split my secret into shares and distribute the shares in such a way that certain specific collection of parties should be able to get back the secret, while the remaining, not remaining, while they are specific blacklisted subset of parties who should not be able to get back the secret.

**(Refer Slide Time: 21:38)**

## Secret Sharing : Real-World Examples



□ Access to the locker:

- ❖ Only if at least two managers come together and enter their password
- ❖ No access, if only a single manager enters its password

$m_1$   $m_2$   $m_3$

So, now, let us try to see where exactly we face this problem of secret-sharing. So, there are plenty of real-world examples which we can model by this abstract problem of secret-sharing. So, consider this problem of granting access to a locker. So, imagine there is a locker which is operated by a password, but the password is kind of shared among 3 managers,  $m_1$ ,  $m_2$  and  $m_3$ .

And the sharing has been done in such a way that, only if 2 managers goes together and enter their respective passwords, then the locker can be opened. But if any one manager tries to open the locker, he or she should fail. So, for instance, if manager 1 just tries to open the locker, he should fail. If manager 2 tries, he should fail. If manager 3 tries, she should fail.

**(Refer Slide Time: 22:35)**

## Secret Sharing : Real-World Examples

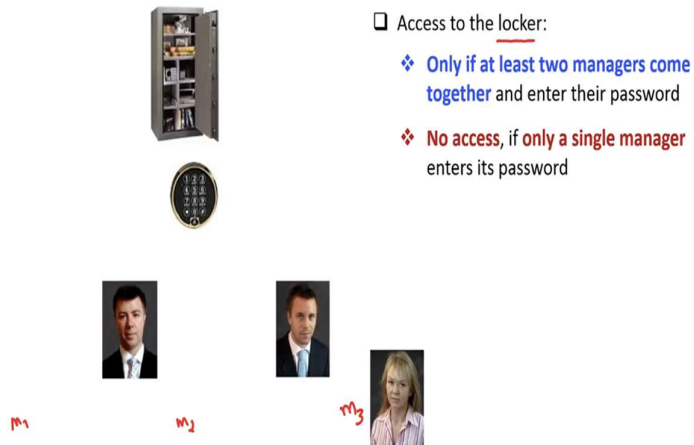


Diagram illustrating a locker and three managers (m<sub>1</sub>, m<sub>2</sub>, m<sub>3</sub>) with a keypad. The locker is shown open, and the keypad is shown below it. The managers are shown as portraits with their respective labels m<sub>1</sub>, m<sub>2</sub>, and m<sub>3</sub> below them.

- Access to the locker:
  - ❖ Only if at least two managers come together and enter their password
  - ❖ No access, if only a single manager enters its password

Whereas, if say the first and the second manager goes together and enter their respective password, the locker should be open.

(Refer Slide Time: 22:42)

## Secret Sharing : Real-World Examples

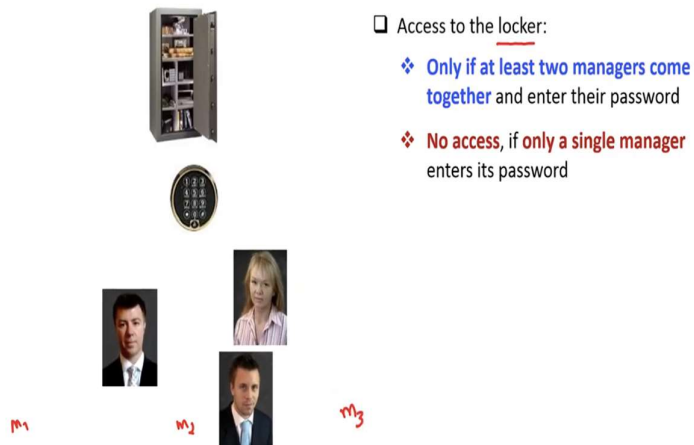


Diagram illustrating a locker and three managers (m<sub>1</sub>, m<sub>2</sub>, m<sub>3</sub>) with a keypad. The locker is shown open, and the keypad is shown below it. The managers are shown as portraits with their respective labels m<sub>1</sub>, m<sub>2</sub>, and m<sub>3</sub> below them.

- Access to the locker:
  - ❖ Only if at least two managers come together and enter their password
  - ❖ No access, if only a single manager enters its password

Similarly, if say, the first and the third manager goes and enter their respective password, the locker should be open and so on.

(Refer Slide Time: 22:50)

## Secret Sharing : Real-World Examples

Access to Russia's Nuclear Weapons in 1990's



President  
          

Prime  
Minister  
          

Defence  
Minister  
          

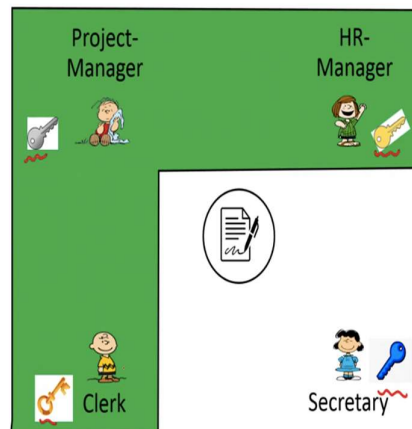
Nuclear Weapons could be accessed ONLY IF AT LEAST  
TWO of the above three entities come together

Now, let us take another example. It is believed according to Times magazine that, in nineties, the access to Russia's nuclear weapon was as follows, was done in the following way: So, there was a password to launch the nuclear weapon, and that password was shared among 3 entities, The President, Prime Minister and Defence Minister. And the sharing was believed to be done in such a way that the nuclear weapon could be accessed only if at least 2 of the above 3 entities come together and enter their respective passwords.

That means, if just President tries to launch the nuclear weapon, he should fail. If Prime Minister tries to launch, and it should fail. If the Defence Minister only tries to launch, it should fail. But if say President and Prime Minister both of them agree and want to launch, they should be able to do that. If Prime Minister and Defence Minister, they agree and want to launch, they should be able to do that. In fact, if all 3 of them wants to launch, they also should be able to do it.

**(Refer Slide Time: 24:06)**

## Secret-Sharing : Real-World Examples

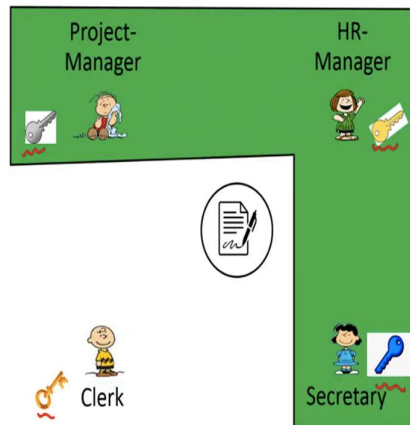


In the same way, imagine that you are, you consider a corporate company where you have, say a bunch of important people. You have, say a project manager, HR manager, secretary and a clerk. And suppose there is a requirement to digitally sign documents, and each of these entity has its own signing key for digitally signing documents. But the arrangement for signing the document has been done in such a way that, no single entity should be able to digitally sign a document on the behalf of the company.

So, for instance, if just project manager wants to digitally sign a document on the behalf of the company, he should fail. If the clerk only wants to sign the document, he should fail, and so on. But say the arrangement has been done in such a way that, if clerk, project manager and HR manager, they agree and decide to digitally sign the document, then only when all 3 of them sign with respect to their individual signing key, that signature should be produced. Even if one of these 3 entities is missing, the signature should not be generated.

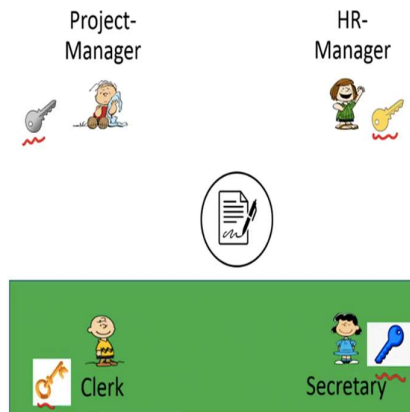
**(Refer Slide Time: 25:27)**

## Secret-Sharing : Real-World Examples



Or, there could be another way to generate the signature. Say for instance, it could be the project manager, HR manager and secretary coming together, and then only they can collectively sign.  
**(Refer Slide Time: 25:40)**

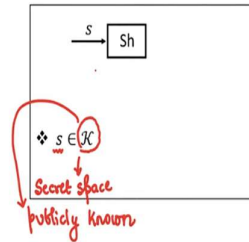
## Secret-Sharing : Real-World Examples



Or, it could be the case that only clerk or secretary are privileged to kind of combine or come together and sign the document.  
**(Refer Slide Time: 25:52)**

## Secret-Sharing : Definition

□ A secret-sharing scheme for access structure  $\Gamma$  over  $\mathcal{P} = \{P_1, \dots, P_n\}$  is a pair of public algorithms (Sh, Rec):



So, those were some examples real-world motivation for studying the secret-sharing problem. So, now, let us come to the formal definition. So, you are given an access structure  $\Gamma$ , over a set of parties,  $n$  parties. So, a secret-sharing scheme; when I say a scheme, it is basically a collection of 2 algorithms; an algorithm for sharing the secret or computing the shares of the secret and an algorithm for reconstructing the secret based on the shares of a collection of parties.

And both these algorithms will be publicly known. Because, as per the Kirchhoff's (26:37) principle which we have seen in the previous course, in cryptography, we never assume that the algorithms are hidden. Algorithms are always assumed to be publicly available. So, let us try to understand first the syntax of our sharing and reconstruction algorithm. So, the sharing algorithm will take an external input, namely, a value of a secret from a bigger space.

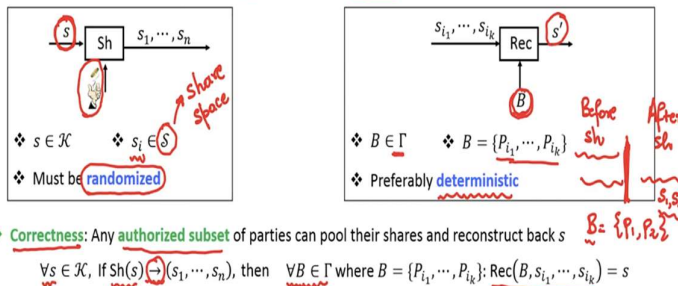
That bigger space will be called as a secret space and denoted by this fancy set  $\mathcal{K}$ . And everyone will know the description of this fancy set  $\mathcal{K}$ . The secret space will be publicly known. But what value from that secret space is used as an input for the sharing algorithm, that may not be publicly known. So, that is the external input for the sharing algorithm.

**(Refer Slide Time: 27:48)**



## Secret-Sharing : Definition

□ A secret-sharing scheme for access structure  $\Gamma$  over  $\mathcal{P} = \{P_1, \dots, P_n\}$  is a pair of public algorithms  $(Sh, Rec)$ :



❖ **Correctness:** Any authorized subset of parties can pool their shares and reconstruct back  $s$   
 $\forall s \in \mathcal{K}, \text{ If } Sh(s) \rightarrow (s_1, \dots, s_n), \text{ then } \forall B \in \Gamma \text{ where } B = \{P_{i_1}, \dots, P_{i_k}\}; Rec(B, s_{i_1}, \dots, s_{i_k}) = s$

❖ **Privacy:** Any unauthorized subset of parties  $B$  "learn nothing" about  $s$  by pooling their shares  
 "The uncertainty about the underlying  $s$  remains the same even after pooling the shares of  $B$ "

And this sharing algorithm should be randomised. So, again, based on whatever we have seen in the previous course, whenever we want to denote a randomised algorithm, I use this symbol. This mean that there are some internal random values which will be generated when this algorithm  $Sh$  will be executed. So, apart from the external input, there will be internal random inputs used inside the algorithm  $Sh$ .

And now, what this algorithm  $Sh$  will do is, based on the internal randomness and the value of your secret  $s$ , it will compute  $n$  pieces or  $n$  values which we call as shares  $s_1$  to  $s_n$ . And each of this share  $s_i$  will be an element from a bigger space, which we call as the share space. Again, I stress that the steps of the algorithm  $Sh$  are publicly known. So, anyone can find out that, if this is the value of  $s$  and if this is the value of candidate randomness, then it can easily compute the shares  $s_1$  to  $s_n$ .

During the actual deployment of the sharing algorithm, the dealer will pick the secret  $s$  and internal randomness which will be generated as part of the instantiation of  $Sh$ , and the shares will be produced. So, that is the syntax of your sharing algorithm. Now, you might be wondering that why we require the sharing algorithm to be randomised. Why cannot it be a deterministic algorithm?

We need it for the purpose of security which will be clear soon. And when I say it is a randomised algorithm, by that, I mean that, if you invoke the sharing algorithm  $Sh$  with the same secret  $s$ , you will be getting different values of the shares, depending upon what is the internal randomness generated during the invocation of your sharing algorithm. So, do not

think that every time you call the sharing algorithm with the input  $s$ , you will get the same value of the shares  $s_1$  to  $s_n$ .

Depending upon your internal randomness, the values of the shares  $s_1$  to  $s_n$  will be different. The *Rec* algorithm will take a subset  $B$  from your access structure. Say for instance that a subset  $B$  consists of  $k$  parties with indices  $i_1$  to  $i_k$  and the external, the other input for this *Rec* function will be the shares corresponding to the parties  $P_{i_1}, P_{i_2}, \dots, P_{i_k}$ . And again, you have the option of either having a deterministic *Rec* algorithm or a randomised *Rec* algorithm; but for most of the secret-sharing schemes, the *Rec* algorithm is deterministic.

So, based on the value of the shares and depending upon whether the shares correspond to an authorised set or unauthorised set, a value will be obtained as an output. Let me denote that output as  $s'$ . Now, what are the properties that we need from this pair of algorithms *Sh* and *Rec*. The first property that we need is the correctness property, which demands that, if your subset  $B$  is actually an authorised subset from the access structure, then the output  $s'$  should be same as  $s$ .

It is like saying the following: If dealer has shared the secret as per the algorithm *Sh*, and later if an authorised subset  $B$  combines or produce their shares; by combining their shares means, applying the *Rec* algorithm. So, if they make their shares available to the *Rec* algorithm, then they should be able to get back the secret. That means, it should not be the case that even an authorised subset fails to get back the secret. That should not be the case.

So, more formally, for every secret  $s$  from the secret space, if the sharing algorithm has generated the shares  $s_1$  to  $s_n$ ; so, what is the interpretation of this notation?  $Sh(s)$  gives the shares  $s_1$  to  $s_n$ . Since my sharing algorithm is a randomised algorithm, I am not using an assignment operator. I am not saying that output of  $Sh(s)$  is equal to  $s_1$  to  $s_n$ , because this is again based on what we learnt in the previous course.

If your algorithm is a randomised algorithm, then  $Sh(s)$ , namely the output of the sharing algorithm with the input  $s$  may generate different shares. So, that is why, I am using this notation arrow. So, the correctness requirement says that, if the sharing algorithm for the secret  $s$  has generated  $s_1$  to  $s_n$ , then, for any authorised subset  $B$ , where the set  $B$  consists of the

parties  $P_{i_1}, P_{i_2}, \dots, P_{i_k}$ , the output of the *Rec* algorithm should be equal to  $s$ , namely,  $s'$  should be equal to  $s$ .

And this should hold for every possible  $B$  in your access structure. Now, let us try to understand the privacy condition which we need from the sharing algorithm. So, recall. We require 2 properties. Authorised subsets, they should be able to get back the secret. Unauthorised subsets, they should not learn anything about the secret  $s$ . Now, what do we mean by learn nothing about the secret.

So, remember, in the last course, we had seen that how difficult it is to formalise a very simple intuition that nothing should be learnt from your ciphertext. We formalized those simple looking intuition in terms of complex games and so on. So, what does it mean when I say that any unauthorised subset of parties learn nothing about  $s$ ? Does that mean they should not learn anything about the magnitude of  $s$ , or does that mean they should not learn anything regarding the first bit of  $s$ , the last bit of  $s$ , or so on?

So, intuitively, the way to interpret this privacy condition is that, whatever was the uncertainty about the underlying  $s$ , that this unauthorised subset of parties  $B$  had prior to the execution of the sharing algorithm, that uncertainty should remain the same. What does that mean? So, imagine that the set  $B$  consists of parties, say  $P_1$  and  $P_2$ . Before the sharing algorithm had been executed, they might have some kind of prior information regarding dealer's secret.

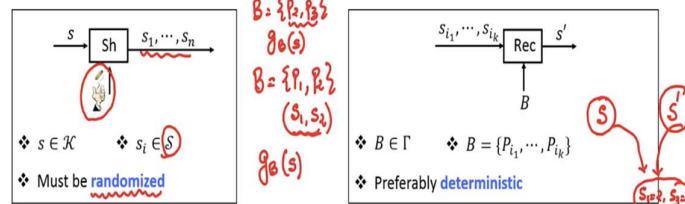
Of course, they will be knowing that the secret is from the space fancy  $\mathcal{K}$ . But they might have some other kind of prior knowledge as well. That, for instance, it could be the case that secret is  $s$ , secret is say this specific value or the secret is going to be this specific value and so on; any kind of prior information. That is the view of the parties in  $B$ , before the sharing algorithm.

And after *Sh*, this collection of parties  $B$ ,  $P_1$ ,  $P_2$  would have the shares  $s_1$  and  $s_2$ . We want to formally capture that whatever they could learn from  $s_1$ ,  $s_2$ , after the execution of the sharing algorithm, that they could have already known before the sharing algorithm would have been executed. That means, whatever was the uncertainty in the left-hand side case, the same uncertainty they have of about the secret  $s$ , even after learning  $s_1$  and  $s_2$ .

**(Refer Slide Time: 36:26)**

## Secret-Sharing : Privacy Definition

□ A secret-sharing scheme for **access structure**  $\Gamma$  over  $\mathcal{P} = \{P_1, \dots, P_n\}$  is a pair of **public algorithms** (Sh, Rec):



❖ **Privacy:** Any **unauthorized subset** of parties  $B$  "learn nothing" about  $s$  by pooling their shares  $B = \{P_1, P_2\}$

≈

The **probability distribution** of the shares of **unauthorized subsets** is independent of underlying shared value

$\forall s, s' \in \mathcal{K}$  where  $s \neq s'$ :  $\{g_B(s)\} \equiv \{g_B(s')\}$  Equivalent

$g_B(s)$ : **random variable**, denoting shares of the parties in  $B$ , when secret  $s$  is shared  
 $g_B(s')$ : **random variable**, denoting shares of the parties in  $B$ , when secret  $s'$  is shared

So, how do we formally state this requirement? The way to formally state this requirement is the following: We will say that the probability distribution of the shares of unauthorized subset of parties should be independent of the secret  $s$ . So, why I am bringing probability distribution? So, remember, I said that my sharing algorithm should be randomized. That means, the shares will take values depending upon what is the internal randomness used during the execution of the sharing algorithm.

That means, the value of the share simply does not depend only on the secret  $s$ . Of course, it will depend with this. That means, the shares will be a function of both the secret as well as the internal randomness. And depending upon the value of the internal randomness, the shares will take different values with different probability. So, that is why, we cannot say that shares will; that means, I cannot say that, for the secret  $s$ ,  $s_1$  can take only one specific value.

$s_1$  could take multiple possible values from the share space, depending upon what is the internal randomness. Similarly,  $s_2$  cannot take a unique value corresponding to a fixed  $s$ . Even though you fix  $s$ , the second value of the share, namely,  $s_2$  can take different values depending upon what is the internal randomness. So, that is why, we have to now consider the probability distribution of the shares, rather than saying the values of the shares.

Because, for the same secret, the shares can take different values depending upon the internal randomness. So, that is why, to formally capture this privacy requirement, we bring this function  $g_B$  or this variable  $g_B$ . So, what is this variable  $g_B$ ? So, imagine  $B$  is a collection of

parties, which is unauthorised. So,  $g_B(s)$  is a random variable which denotes the shares corresponding to the parties in  $B$ , when the secret  $s$  is shared by your sharing algorithm.

So, again, if your  $B$  is equal to, say  $P_1, P_2$ , then the shares learned by  $P_1$  and  $P_2$  are  $s_1$  and  $s_2$ . So, since I said that  $s_1$  and  $s_2$  can take different values for the same secret  $s$ , depending upon the internal randomness,  $s_1$  and  $s_2$  can have different values. So, I can denote, I can use a variable which denotes the values of, combined value of  $s_1$  and  $s_2$ . So, that random variable is my  $g_B(s)$ .

If my  $B$  would have been, say  $P_2$  and  $P_3$ , then my  $g_B(s)$  will denote that what could be the possible values of the shares  $s_2, s_3$ , when the secret  $s$  would have been shared by the sharing algorithm  $Sh$ . So, that is a random variable,  $g_B(s)$ . And in the same way,  $g_B(s')$  denotes the random variable; it denotes a variable which stands or which captures the shares of the parties in the unauthorised collection  $B$ , when the secret  $s'$  is shared.

And again, as I said that if  $s'$  is shared, then it would not be the case that the values of the shares corresponding to the parties in  $B$  will be the same shares. It will take different values depending upon internal randomness. So, that is why we bring a random variable  $g_B$  for  $s'$ . So, now, the privacy condition is formally the following: We demand that for every possible pair of secrets  $s$  and  $s'$  from the secret space, where  $s$  and  $s'$  are different secrets, the probability distribution, so, this notation,  $\{g_B(s)\}$ , it denotes the probability distribution.

Namely, the shares corresponding to the parties in  $B$  when the secret  $s$  would have been shared with whatever probability. And on your right-hand side, you have the probability distribution for the shares of the parties in  $B$  corresponding to the secret  $s'$ . And this notation  $\equiv$  which looks like an equal symbol, it is not an equal symbol, it denotes equivalent. That means, we want here that, with whatever probability the shares of the parties in  $B$  can occur for the secret  $s$ , the same shares could occur with same probability even if the secret  $s'$  would have been shared.

So, that means, if, say for instance, if I consider  $B$  to be the parties  $P_1, P_2$ ; and say, if they see the shares  $s_1 = 2$  and  $s_2 = 3$ ; I am just taking an abstract example. Then, with equal probability, this shares  $s_1 = 2$  and  $s_2 = 3$  could occur for the secret  $s$ . And with same

probability, the share  $s_1 = 2$  and  $s_2 = 3$  should have occurred for the secret  $s'$  as well, if  $s$  and  $s'$  would have been independently secret shared by the algorithm  $Sh$ .

If this property is ensured by your sharing algorithm, then this is equivalent to saying that, if the parties in  $B$  see the value of their shares, then they cannot pinpoint or they cannot distinguish whether they are seeing the shares for the secret being  $s$ , or whether they are seeing the shares corresponding to the secret being  $s'$ . And hence, for them, it could be as if both  $s$  or  $s'$  would have been shared. And hence, they learn nothing about the underlying secret.

(Refer Slide Time: 43:11)

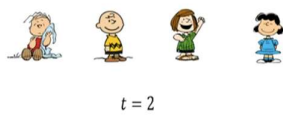
### t-out-of-n Secret Sharing : Definition

=  $\rightarrow$  parameter

- A special case of general secret-sharing

**Access Structure**

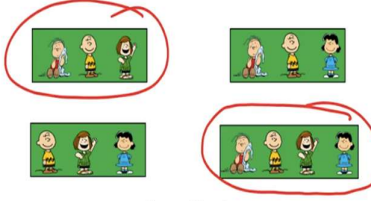
❖ All subsets of  $t + 1$  or more parties



$t = 2$

**Forbidden Structure**

❖ All subsets of  $t$  or a smaller number of parties



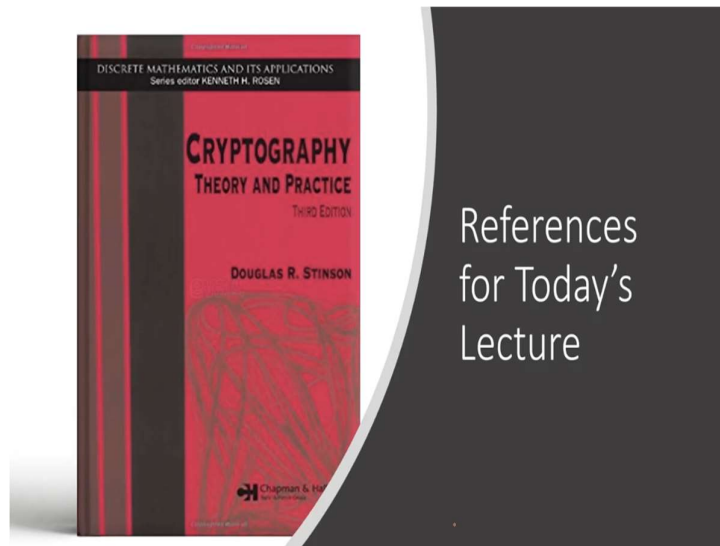
Access Structure

- Specialized efficient schemes can be designed tailor-made for the **threshold setting**

Now, let us focus on a special case of secret-sharing, which we call as threshold secret-sharing or  $t$  out of  $n$  secret-sharing. And here,  $t$  will be an input parameter given to you. And this is a special kind of secret-sharing scheme where the access structure consists of all subsets of  $t + 1$  or more parties. That means, your minimal authorised subsets will have cardinality  $t + 1$ .

And complementarily you will have the forbidden structure where each maximal unauthorised subset will have cardinality  $t$ . Namely, any subset of  $t$  or small number of parties should fail to learn the secret. So, if I take the case of  $n = 4$  and  $t = 2$ , then it means that you have these subsets as the authorised subsets. So, subset  $P_1, P_2, P_3$ ;  $P_1, P_2, P_4$ ;  $P_2, P_3, P_4$ ; and of course, if I include any additional party along with any 3 parties, that also constitute an authorised subset. But I would not be explicitly listing it out in my access structure. Because, in my access structure, remember, I focus only on the minimal authorised subset. And why we are interested in this  $t$  out of  $n$  secret-sharing? Because, looking ahead, we can design specialised efficient schemes which are tailor made only for this specific case of threshold setting.

**(Refer Slide Time: 45:05)**



So, with that I end today's lecture. There are plenty of references to understand the problem of secret-sharing. I used the textbook by Douglas Stinson as for the purpose of reference. Thank you.