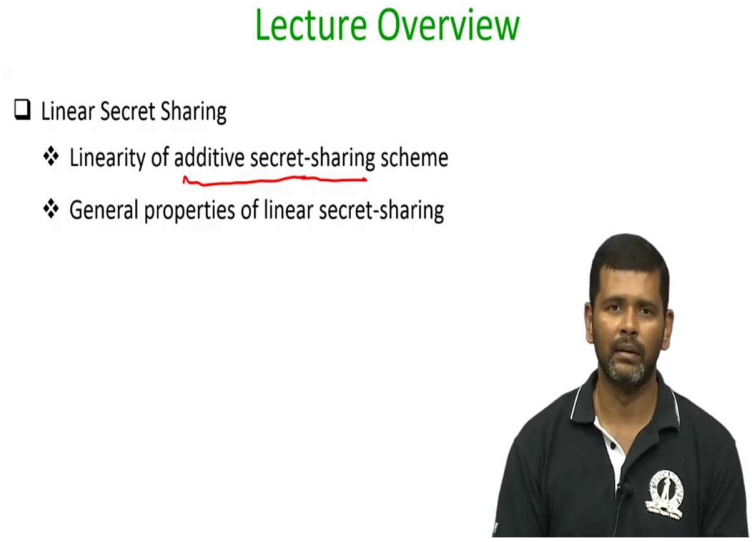
Secure Computation - Part I
Prof. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology, Bangalore

Module - 3
Lecture - 13
Linear Secret-Sharing Contd.

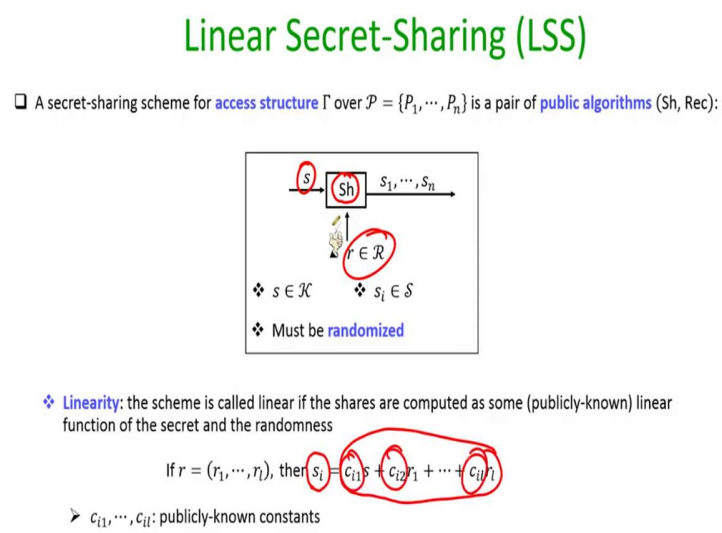
(Refer Slide Time: 00:34)



Hello everyone. Welcome to this lecture. So, in this lecture, we will continue our discussion on linear secret-sharing. So, in the last lecture, we had seen the linearity property of Shamir secret-sharing and some magic associated with it. Namely, we have seen that, even without revealing the underlying secret-shared values, we can perform, we can compute any linear function of those secret-shared values.

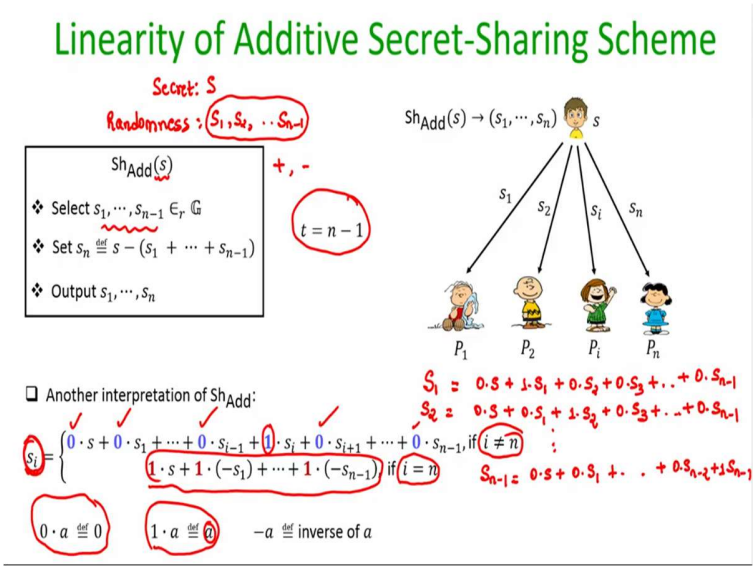
Namely, we can compute the shares of the result of performing or computing some linear function on the underlying secret-shared values. Now, in this lecture, we will see that even our additive secret-sharing scheme, which is an n - 1 out of n secret-sharing scheme, also satisfies the linearity property. And then, we will wrap up with some general properties of any linear secret-sharing scheme, which may not be just additive or Shamir secret-sharing, but it could be any linear secret-sharing scheme.

(Refer Slide Time: 01:29)



So, just to recap. When do we say that our secret-sharing scheme is a linear secret-sharing scheme? If the shares are computed in my sharing algorithm as per some linear function of the secret and the randomness. Namely, each share s i should be computed as some linear combination of the secret and the individual components of the randomness, as per some publicly known constants.

(Refer Slide Time: 02:03)



So, now, let us recall the additive secret-sharing, where all the operations are performed over the group. To share a value s, the first n - 1 shares are picked uniformly at random from the group, and the last share is set in such a way that the summation of n shares together give you the secret s. And here, the degree of sharing is n - 1. The sharing satisfies the property that, only when the entire set of n parties come together, they constitute an authorised set.

Even if 1 party is missing and the group of n - 1 parties, even if they are computationally unbounded, they learn absolutely nothing about the underlying secret. So, now, let us see whether the additive secret-sharing scheme satisfies the linearity property. So, this might look like a complicated expression, but it is not. So, can I express the share s_1 as a linear combination of the secret and the randomness?

So, remember, my secret in the additive secret-sharing scheme is s, and the internal randomness is basically the first n - 1 shares, namely, s_1, s_2, s of n - 1. Depending upon the value of this randomness, namely, the first n - 1 shares, the value of the nth share is computed. So, I can imagine, I can rewrite this share s_1 as a linear combination of the secret s and the randomness as follows:

I can say that s_1 is 0 times s; that means, it has got nothing to do with s; plus 1 times s_1 plus 0 times s_2; and like that 0 times s_3, no dependency on s_3; and like that, 0 times s of n - 1. I can interpret s_2 as, it has got 0 dependency on s, 0 dependency on s_1; it depends on s_2, namely, the linear combiner is 1 here. And on the remaining components of the randomness, it has no dependency.

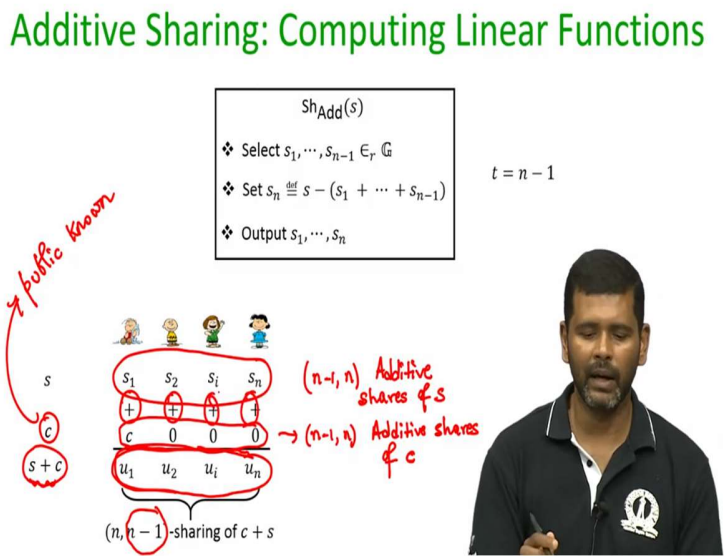
So, that is why the linear combiners are 0 here. And continuing like this, I can say that s of n - 1 is 0 times s, 0 times s_1, and like that, 0 times s_{n-2}; but it depends on s_{n-1}. So, that is the interpretation of this first part of this s_i. So, whenever i is any index different from n, I can say that s_i depends only on itself; that is why the linear combiner is 1; and all other linear combiners are 0.

But when it comes to the nth share, nth share depends not only on s, it depends also on s_1, s_2, s_{n-1}. That is why I am having s of n of this one. So, now, since all the operations are performed over the group, and in the group, I only have the plus operation, and the minus should be interpreted as the additive inverse; but in my expression of s_i, I have bought the dot operation.

So, by the dot operation, I mean here the following: If I say 0 times a, I mean the additive identity element 0; and 1 times a, I mean here the element itself; and minus a here means the additive

inverse of a. So, that means, now I can express each share as a linear function of the secret and the randomness.

(Refer Slide Time: 06:06)



So, now, similar to the case of Shamir secret-sharing, we will see that we can perform linear operations on the underlying shared value, even for the additive secret-sharing scheme, without knowing the value itself. So, imagine that there is some value s which is additively shared. This is a vector of n - 1, n additive shares of s, where each party P_i has the ith component. And imagine that there is some value c in the group.

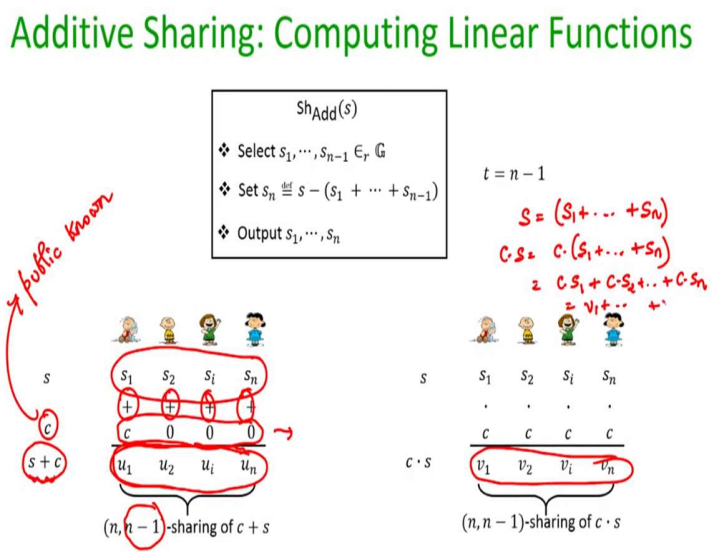
And I ask only the first party; I means, as part of the protocol step or some instruction, only the first party adds c to s_1, namely, its share of s; and remaining parties do not change their shares of s. Suppose, together, collectively the n parties perform this operation as a protocol, as an algorithm. Now, they will obtain some values, because the plus operation satisfies the closure property, the resultant elements also will be group elements.

Now, what can I say about this vector of new elements? Is it an arbitrary vector of n group elements? No. If you see closely here, the vector of these values u_1, u_2, u_i, u_n, if I sum these n values, basically I will get the value s + c. Why so? Because this vector of values; here the first component is c and remaining components are 0. It can be considered as an n - 1, n additive shares of c, where everyone knows the value c; that is publicly known, remember.

And now, if I add these 2 vectors, I will obtain another vector such that the summation of all the n pieces will give you s + c. And indeed, the threshold of the sharing for the new vector is n - 1. Only when all the n u values are available, namely, when all the n parties come together and make their u values public, then only you can reconstruct the value s + c. Even if 1 of the u values is missing from this vector; that means, if an unauthorised subset of parties consisting of n - 1 or less number of shareholders come together, they cannot figure out what is the value of s.

For them, it could be any s plus this public c which has been shared among the n parties. So, that is why, this plus operation can be performed on the shares itself, without knowing the underlying secret-shared value.

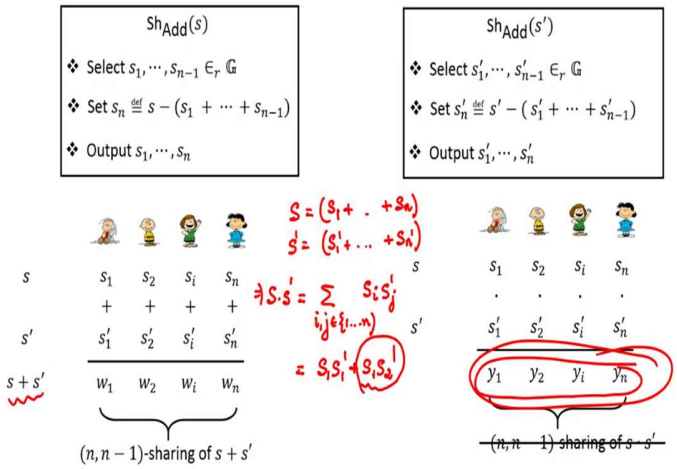
(Refer Slide Time: 09:22)



In the same way, what if each party multiplies its share of s with this public constant c, where c is now a publicly known group element? They will now obtain some element from the group itself. And now, it is easy to see that this vector constitutes additive secret-sharing of the element c times s; because, your s was s_1 up to s_n; and c time s is basically c times s_1 up to s_n.

(Refer Slide Time: 10:21)

Additive Sharing: Computing Linear Functions

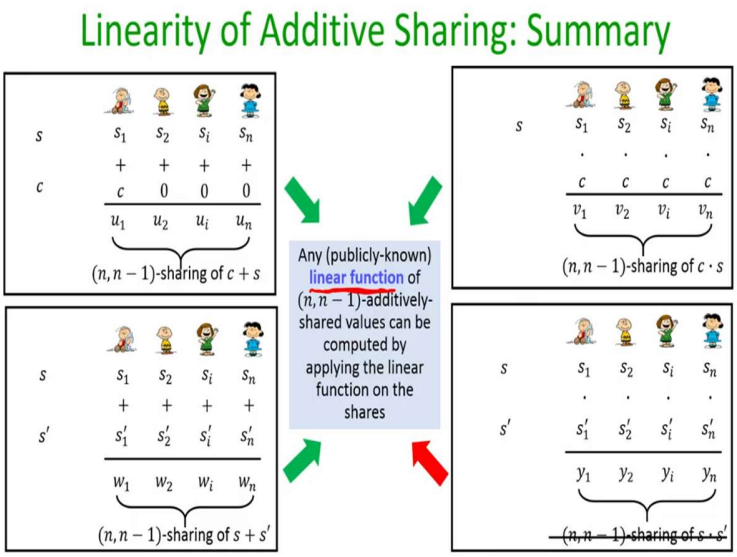


On the other hand, if you have 2 values s and s' which are additively shared, and then we ask the individual party to add its respective share of s and s', then even without knowing s and s', each party ends up getting its respective share of s plus s'. But, what if I ask each party to multiply its respective share of s and s'? Will the resultant vector of values constitute an additive secret-sharing of s times s'?

The answer is no; because, your s was the summation of n values, your s' was the summation of n values. That means, if you want to compute a vector of additive shares for s times s', then it will be consisting of n square sum s. Namely, it will be the summation of all i j ranging from 1 to n, and s_i times s'_j. We have not obtained all the individual terms here, right?

So, if I expand here, then it is s_1 times s'_1 plus s_1 times s_2 prime and so on. So, terms like this are not computed in this vector. So, that is why this resultant vector of values does not constitute an additive secret-sharing of s times s'.

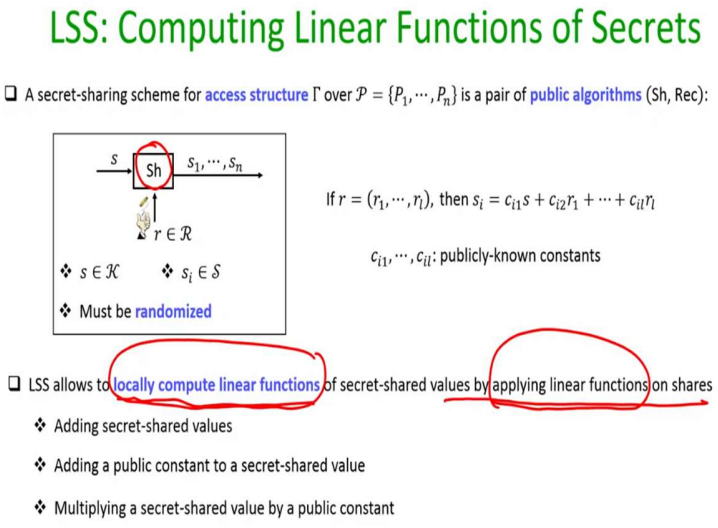
(Refer Slide Time: 12:17)



So, again, similar to the case of Shamir secret-sharing, what we have seen here is that, if there is some linear operation, namely adding a public constant or multiplying some public constant with some secret-shared value, it can be performed on the shares itself, without knowing the underlying secret-shared value. But when it comes to multiplying 2 secret-shared values, that cannot be performed just by multiplying the individual shares of the 2 secret-shared values.

That means, in summary, we can say that any publicly known linear function of additively secret-shared values can be computed by applying the linear function on the shares itself, and without disclosing the inputs of the linear function as well as the output of the linear function.

(Refer Slide Time: 13:10)



So, now, we had seen the linearity property with respect to 2 specific secret-sharing schemes, namely additive secret-sharing and Shamir secret-sharing. Now, coming back to an abstract linear secret-sharing scheme. So, imagine if your sharing algorithm is a linear function, it computes the share as a linear function of the secret and the randomness, then this linear secret-sharing allows to locally compute linear functions, as we have demonstrated with 2 concrete secret-sharing schemes.

So, you can compute linear functions of the secret-shared values by applying the linear function on the shares itself. And in the process, you learn the outcome of the linear function also in secret-shared fashion. That means, neither the inputs of the linear functions were available to the respective parties, but rather each party had shares of the input of the linear function.

And by performing or by applying the linear function on the shares of the input of the linear function, they obtain shares of the output of the linear functions. And some of the interesting linear functions which can be computed over the shares are adding secret-shared values, adding a public constant to a secret-shared value, multiplying a secret-shared value by a public constant and so on. So, this is a very powerful property, this property of linear secret-sharing; because, later on, it will be useful tremendously when we will design MPC protocols. Thank you.