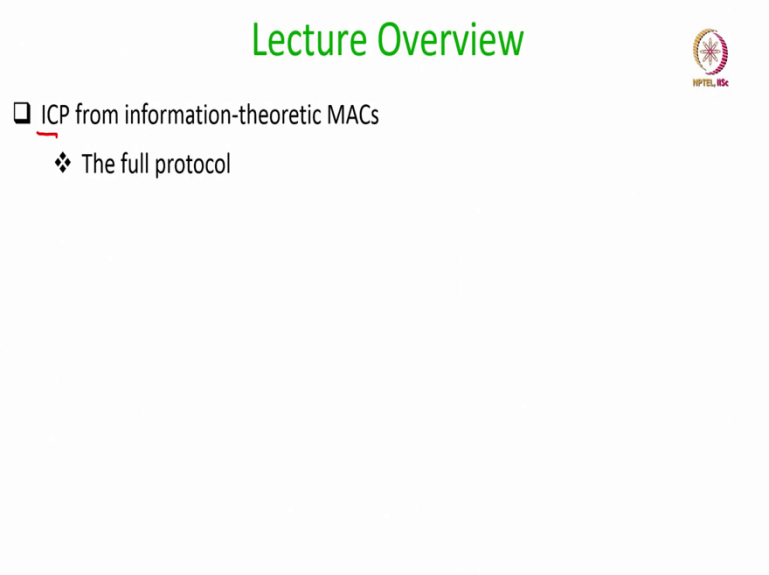
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

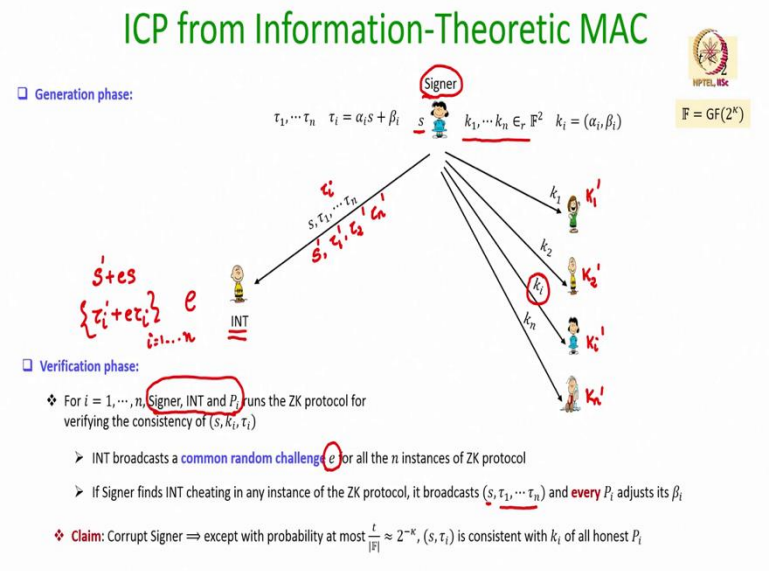
Lecture - 56
IC Protocol

(Refer Slide Time: 00:29)



Hello everyone welcome to this lecture. So, in this lecture we will complete the description of the IC Protocol which will take care of both honest signer as well as a potentially corrupt signer.

(Refer Slide Time: 00:37)



So, we will first begin with the protocol with the honest signer which we had discussed in the earlier lecture and then we will augment it with an intermediate verification phase. So, to sign the value s in the generation phase, signer is going to pick n independent mac keys where the ith key will be for the ith verifier. And to give the signature, signer computes tags for the data s with respect to each of these keys. And now the data and tags on the data s is are given to intermediary and individual mac keys are given to the individual verifiers.

Now, we introduce here the verification phase to verify whether a potentially corrupt signer has distributed consistent data to intermediary and honest verifiers. And for this we run the zero knowledge consistency check which we had discussed in the earlier lecture. And one instance of that zero knowledge check is executed with respect to each of the individual verifier.

So, you have n such instances running, where the ith instance is to verify whether the ith tag here which is given to the intermediary is the right tag with respect to the ith key ki given to the ith verifier. Now if you recall the zero knowledge consistency check, then in one instance of the zero knowledge consistency check, INT is supposed to make public a random challenge or a random linear combiner e.

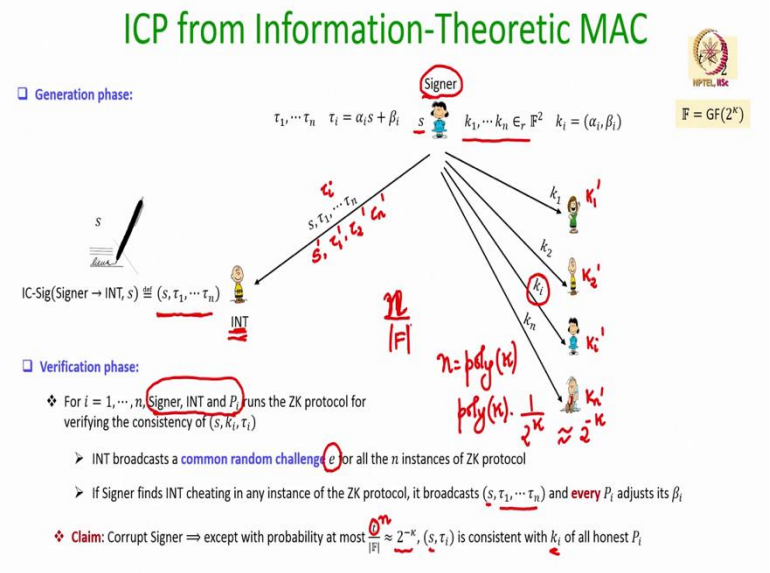
So, you might be wondering that since there are n such instances of zero knowledge running, whether INT should make public n number of random challenges, random linear combiners and the answer is no. What INT can do is the following. Once signer has given

the tag with respect to the keys of all the n individual verifiers, INT can select and broadcast a single random challenge e, which can be used for all the n instances of the zero knowledge consistency check ok.

So that means, as part of the zero-knowledge consistency check, INT will be also receiving s' and another tuple of tags τ_1', τ_2', τ_n'. And each individual verifier will also receive an auxiliary mac key k_i'. And then INT will make public a single random combiner e. Along with that it will make public the value $s' + es$. And then it will make public the corresponding linear combination of the ith tags ok. Then as part of the consistency check signer will check whether INT has behaved correctly or not.

If signer finds that INT is cheating in any of the instances of the zero-knowledge protocol, then it can make s public and all the tags public. And accordingly, every verifier will adjust the β-component of its key retaining the α-component, so that the new key becomes consistent with respect to the ith tag on the data s which has been mac public by the signer ok. And the rest of the protocol remains the same. So now, we can claim here the following we can claim the non repudiation property, we can claim here that if the signer is corrupt ok.

(Refer Slide Time: 05:45)



If the signer is corrupt, then except with probability t over the field size which will be asymptotically same as $2^{-\kappa}$, for every honest verifier Pi, whatever key it has at the end of the protocol, with respect to that key, INT will have the right tag on the data s. And this

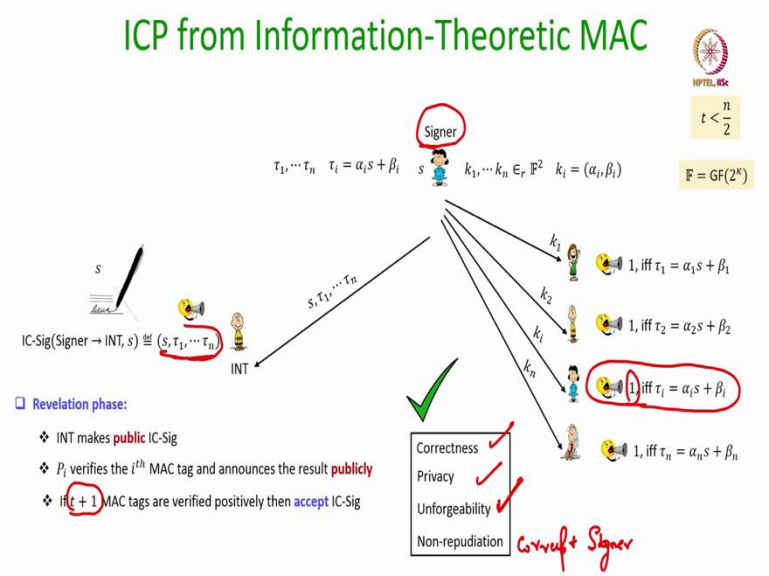
comes from the zero knowledge consistency check executed with respect to INT and the ith verifier ok. Now you might be wondering why in the numerator t is coming, ok. Actually it should be n over |F|, not t over |F|.

Because it could be the case that say all the verifiers are honest ok, let us take that case and say the signer is corrupt and it has distributed inconsistent information to INT and the verifiers. So, that inconsistent information could be with respect to say INT and the first verifier or with respect to INT and the second verifier or with respect to the INT and the nth verifier.

If it is between INT and the ith verifier, then we know that except with probability one over F, the signer will be caught, caught in the sense that either it will be discarded or it will be forced to make the ith key public. So that INT can adjust its ith tag to be consistent with whatever key signer has made public for the ith verifier. So, since there could be up to n honest verifiers, by applying the union bound, the maximum probability with which a corrupt signer can bypass the protocol is n over F, ok, and n over F will be still a negligible quantity in kappa, because n will be polynomial function in your security parameter kappa. So, any polynomial function in the security parameter multiplied with an exponentially small function in the security parameter will be still an exponentially small function in your security parameter.

So; that means, at the end of the verification phase it will be guaranteed that except with a negligible probability, whatever data INT has it is consistent with respect to the honest verifiers in the system. And the data which is held by INT at the end of the verification phase after adjusting all tags etcetera is considered as the IC signature of the signer given to it.

(Refer Slide Time: 08:55)



Now, the revelation phase remains the same as it was for the case of honest signer. Namely INT will make public the entire signature, which in turn includes the data s and tags on the same data s with respect to the keys of all the individual verifiers. Now to check whether INT has made public the right data, every individual verifier will locally verify only the tag corresponding to its key.

So, the ith verifier will check the tag out of these n tags with respect to it is key and it will either make public the value 1 or o depending upon the verification is successful or not. And then to finally, make a decision whether to accept the intermediry's message, we check whether there are at least t + 1 parties who have responded positively.

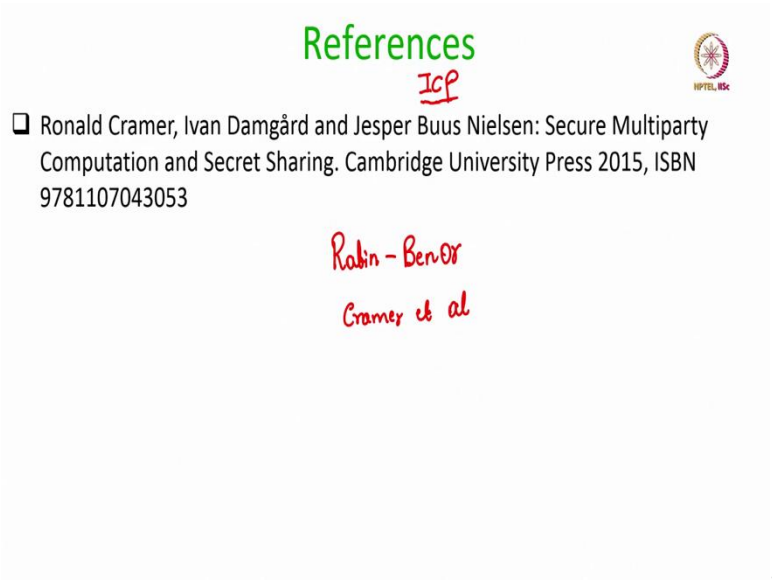
That means they have individually verified the tag, the corresponding tags, on this data s which have been made public by INT. If they have done so, then we can take the data s as accepted. Otherwise we decide to reject the signature and now you can see that all the required properties of ICP are satisfied. So, we had already seen the correctness, privacy and unforgeability properties, all these are with respect to an honest signer.

Now with respect to a corrupt signer, even the non-repudiation property is achieved, because of the verification phase which we have introduced in the system. Namely, if the signer is corrupt and it has distributed inconsistent mac tags with respect to the keys of the individual honest verifiers, then except with a negligible probability it will be caught in

the system and either will be discarded or the signer will be forced to make public either the tags or the keys.

And once it makes public the tags or keys, then INT and verifiers can behave accordingly by adjusting their respective tags and keys, which will ensure that at the end of the verification phase, their data and tags are consistent with respect to the keys of the honest verifiers in the system ok.

(Refer Slide Time: 11:37)



So, that completes the description of the information checking protocol. Information checking protocol has been a very fundamental primitive for statistically secure MPC. There are other ICP, the ICP that I had discussed in this lecture is taken from this textbook, but you can refer to the first protocol on statistically secure MPC, namely the classic paper by Rabin Ben-Or to get other types of ICP. You also have a paper by Cramer et al, which also presents a different form of ICP and so on.

Thank you.