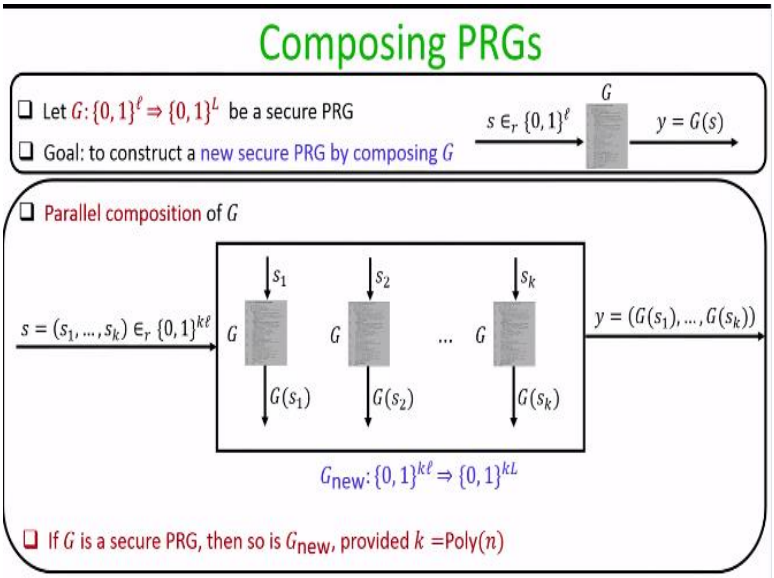
Foundations of Cryptography
Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
Indian Institute of Technology – Bangalore

Lecture – 9
Composing PRGs

Hello everyone, welcome to this lecture. In this lecture, we will continue our discussion on pseudorandom generators, namely we will see how to compose PRGs.

(Refer Slide Time: 00:39)



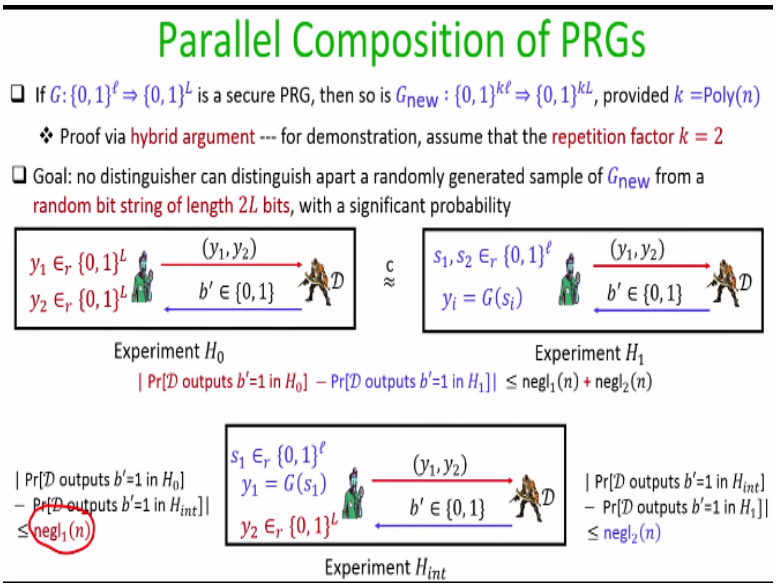
This is a very popular operation which we perform on PRGs and by composing PRGs basically we want to increase the input size and output size of PRG. That means, imagine you are given a secure PRG, forget for the moment the steps of the algorithm G and algorithm G basically takes an input of size little l bits and produces an output of size big L bits and now our goal is to basically compose many independent executions of the algorithm G, namely here we will consider the parallel composition of G.

In our future discussion, we will also consider the serial composition of G. So, by doing the parallel composition of the algorithm G, our goal is to design a new random number generator which I denoted by G new, which basically takes an input of size kl bits and it should produce an output of size k times big L bits. So, you can imagine that this algorithm G new now takes k blocks of inputs where each block is of size little l bits and each of these blocks of little l bits are uniformly random.

Internally what this algorithm G new is doing is it is running the algorithm G the existing algorithm G on the first block, independently it is running another copy of the algorithm G on the second block and like that independently it is running a kth copy of the algorithm G with the last block as the input. It simply concatenates the outcome of each of these independent invocations of the algorithm G and that is defined to be the outcome of this algorithm G new and that is how you actually parallely composing the algorithm G.

Now, we want to prove here that if the number of copies or the number of times we have composed this existing algorithm G, namely k, is some polynomial function of your security parameter n and if your existing algorithm G is a secure PRG as per any of the definitions either indistinguishability based definition or next bit predictor, then we want to prove that the new algorithm G new which we have obtained by composing the PRG is in parallel is also a secure PRG.

(Refer Slide Time: 02:52)



For this, we are going to introduce a new proof strategy which we call as hybrid argument, and this is a very popular proof strategy used extensively in modern cryptography primitives. For purpose of demonstrating this hybrid argument, I will consider the repetition factor to be k = 2, do this is just for simplicity and later we will see the case for a generic k, where k is any polynomial function of the security parameter, right. So, if I consider k = 2 that means, my algorithm G new now consist of 2 independent copies, 2 parallel copies of existing algorithm G.

I want to prove that this algorithm G new is a pseudorandom generator, and I want to use the indistinguishability based definition. So, my goal is to show that there exist no polynomial time distinguisher who can distinguish apart a uniformly random sample generated by this algorithm G new from a uniformly random sample generated by running a truly random generator, which outputs uniformly random strings of length 2L bits, right. So, for this, consider 2 different experiments, which are denoted by H0 and H1.

In both these experiments, the challenge for the distinguisher is a sample consisting of 2 blocks of big L bits, which are denoted by y1 and y2. In both the worlds, the distinguisher has to find out the way this sample y1, y2 has been generated. So, in experiment H0, the first part of the sample as well as the second part of the sample are both uniformly random strings of length big L bits and that is how you can imagine a challenge sample for the distinguisher would have been generated.

If uniformly random string of length 2L bits would have been given as the challenge for the distinguisher. Whereas in experiment H1, both parts of the challenge namely y1 and y2 are generated by invoking the existing algorithm G on uniformly random seeds s1 and s2 and by running the algorithm G independently twice. So you can imagine that this experiment H1 is the version of the indistinguishability based experiment if the distinguisher would have participated in the indistinguishability based experiment.

The sample whichever I would have been given to D would have been generated by our algorithm G new. Now, our goal is to prove that both these versions of the experiment are computationally indistinguishable, which I denote by this notation. So, this notation means that these 2 versions of the experiments are computationally indistinguishable and what I want to prove here is that if my existing algorithm G is indeed a secure PRG as per the notion of indistinguishability based experiment, then with almost equal probability D would have output the same output in experiment H0 as well as an experiment H1.

That means the distinguishing probability or the distinguishing advantage of my distinguisher for any polynomial time, distinguisher is upper bounded by a negligible function. That is what I want to prove when I say that I want to prove my algorithm G new is a secure PRG right. Now, it turns out that we cannot directly prove or we cannot directly reduce the security of the algorithm G new to the instance of the security of the existing algorithm G because the

algorithm G produces only one sample of size big L bits, whereas in algorithm G new you are actually invoking your existing algorithm G twice.

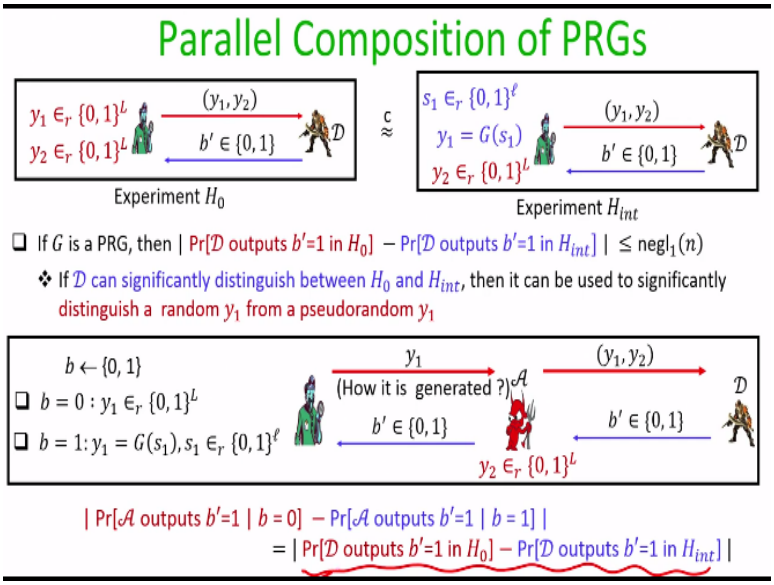
So, to prove the computational indistinguishability of the experiment H0 and H1, what I am going to do is I am going to introduce an intermediate experiment which I denote as Hint and on a very high level, this intermediate experiment is somewhat intermediary between H0 and H1. That means, here also the distinguisher will be given a sample consisting of 2 blocks of size big L bit, big L bits, but the difference here is that the first part of the sample which would have been generated by running the algorithm G on a uniformly random input.

Whereas the second part of the sample which would have been generated by running a truly random generator, and now, what we are going to prove is, we will prove 2 different claims. The first claim will be we will claim that if my existing algorithm G is indeed a secure PRG, then both experiment H0 and experiment Hint are computationally indistinguishable. That means, any polynomial time distinguisher is going to output the same output bits in both versions of the experiment whether it is H0 or H1 except with a negligible probability, which I denote by negligible 1.

In the same way, I am going to prove that if my existing algorithm G is a secure PRG, then my intermediary experiment, Hint and H1 are computationally indistinguishable from the viewpoint of any polynomial time distinguisher. That means, with almost identical probability of any polynomial time, distinguisher is going to output the same output bit irrespective of whether it is participating in experiment H1 or whether it is participating in experiment H2 except with some negligible function, which I denote by negligible 2.

Now, if I prove these 2 claims, then by summing these 2 distinguishing probabilities, I can end up showing that my experiment H0 and H1 are also computationally indistinguishable, namely the probability with which the distinguisher could distinguish apart whether it is participating in experiment H0 or whether it is participating in experiment H1 will be upper bounded by the summation of 2 negligible probabilities, and from the closure property of the negligible probability function, we come to the conclusion that the sum of 2 negligible functions is also a negligible probability.

(Refer Slide Time: 09:22)



So, let us proof the first claim. That means, we want to prove that if G is a secure PRG, then no polynomial time distinguisher cannot distinguish apart whether it is participating in experiment H0 or whether it is participating in experiment Hint except with some negligible success probability. The intuition behind this claim or the statement is that if we have a polynomial time distinguisher who can significantly distinguish apart whether it is participating in experiment H0 or whether it is participate in experiment Hint.

Then using that distinguisher we can design another distinguisher who can distinguish apart a uniformly random y1 from a pseudorandom y, and let us firmly establish this intuition. So, imagine for the moment you have a distinguisher D who can distinguish apart whether it is participating in an instance of experiment H0 or whether it is participating in an instance of an experiment Hint.

Now, using this distinguisher, our goal is to design another polynomial time distinguisher which I denote by A whose goal is to distinguish apart a uniformly random sample generated by an algorithm G versus a truly random sample of size big L bits generated by a truly random generator, right. So, the algorithm A participates in an instance of my indistinguishability based definition or experiment for the PRG where it will be thrown a challenge y1 of big L bits and the challenge for the algorithm A is to find out whether y1 is generated by the algorithm G or by a truly random generator.

Now, what the algorithm is going to do is algorithm is going to take the help of algorithm D, right. Before going into how exactly algorithm A takes the help of the algorithm D, let me just recall that as per the syntax of our indistinguishability experiment, the way sample y1 would have been generated is as follows. The verifier of the indistinguishability based experiment would have tossed a coin, if the coin would have output 0, then the sample y1 is generated by a truly random generator.

Whereas if the coin is 1, then the sample y1 is generated by running the algorithm G on a uniformly random input. The challenge for our algorithm A is to find out what exactly b is, whether b = 0 or whether b = 1. Now, what the adversary is going to do is it itself is going to generate a uniformly random string which I denote by y2 of size big L bits and it produces a new challenge or a new sample for the distinguisher D by concatenating the challenge y1 which was thrown to A with the sample y2 which it has generated uniformly randomly.

Now, what exactly is happening here? Right. So, let us pause here for a moment. If you see the way adversary A has done the computation here, if the sample why y1 would have been generated uniformly randomly, then y1, y2 would have looked as if it is a challenge that the adversary D would have expected by participating in the experiment H0 because in the experiment H0 both y1 as well as y2 are generated uniformly randomly.

That means in this reduction, if the sample y1 which is thrown as a challenge to the adversary is generated by running a truly random generator, and if it is concatenated by a truly random sample, another independent sample of size big L bits, then y1, y2 would have looked as a challenge which the adversary D would have expected by participating in the experiment H0, right. On the other hand, if the sample y1 which is thrown as a challenge to the adversary is generated by running a pseudorandom generator G.

Then this y1 concatenated with a uniformly random sample y2 would look a challenge for the distinguisher, which the distinguisher would have expected by participating in an instance of the experiment Hint. Because in this intermediary experiment, the first part of the sample is generated by running a pseudorandom generator, whereas a second part of the sample the challenge sample is uniformly random. Now, our adversary A does not know whether it has actually forwarded a sample as per the experiment H0 or whether it has forwarded a sample as per the experiment Hint to the distinguisher.

It is a distinguisher D who can actually identify whether y1, y2 it is saying is generated as per H0 or as per Hint. That is what I mean when I say that we have a distinguisher who can significantly distinguish apart whether it is participating in an instance of H0 versus an instance of experiment Hint, right. So, whatever is the case based on the sample y1, y2 which is given to the distinguisher, distinguisher is going to output a bit, say b dash, which indicates whether the sample is generated as per experiment H0 or whether it has been generated as per Hint.

Now, depending upon the output of the algorithm D, what A is going to output? It is going to produce the same output as D going to produce. That means if D says that the sample that it is seeing is generated as per experiment H0, then A labels the sample y1 as if it is generated by a truly random generator, whereas if the distinguisher D says b dash = 1, that means if it says that **y** the sample y1, y2 is generated as per the intermediary experiment, then the adversary A says that the sample y1 is generated as per the pseudorandom generator.

So, now let us calculate the distinguishing advantage of the algorithm A, which we have constructed using the existing distinguisher D. So, let us first calculate the probability that our algorithm A outputs or labels uniformly random sample y1 as the outcome of a pseudorandom generator. That means, we want to calculate the probability that A outputs b dash = 1 even though b = 0, and the claim is this is exactly the same probability with which our distinguisher D is going to output b dash = 1 in the experiment H0, and this is because if b = 0, right.

If b = 0, then we are in the case where y1 would have been generated by a truly random generator and that y1 concatenated by y2 would have created a sample for D as per the experiment H0 zero. Namely, the view of the distinguisher would have been exactly the same as it would have by participating in the experiment H0, right? Whereas the probability that our algorithm A outputs b dash = 1 given b = 1, that means it outputs the sample y1 as it labels the sample y1 as the output of a pseudorandom generator.

Given that it was indeed generated by a pseudorandom generator is exactly the same with which our distinguisher D, the existing distinguisher D would have output b dash = 1 by participating in an instance of the experiment Hint because if b = 1, that means the challenge

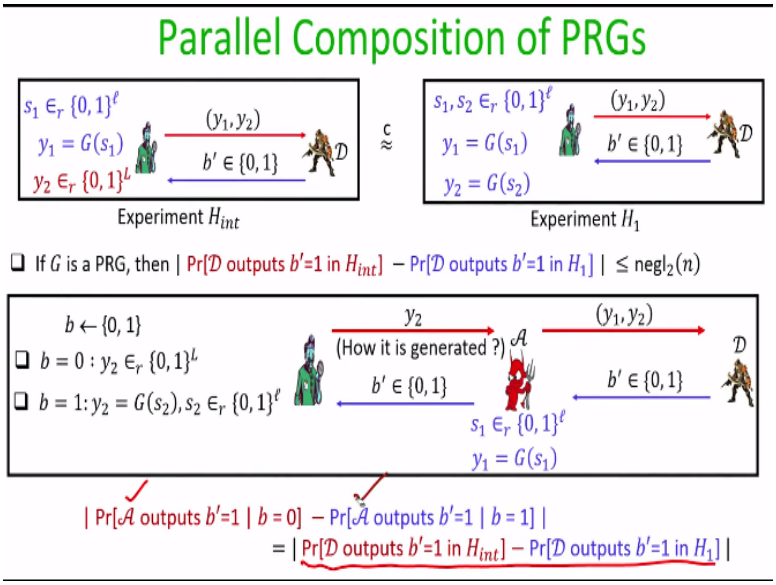
sample y1 for A generated by a pseudorandom generator, then that pseudorandom sample generated concatenated by a truly random sample would look like a sample that D expects by participating in an instance of the experiment Hint.

So, with whatever probability D would have output b dash = 1 in the experiment Hint, with exactly the same probability our adversary A is going to output b dash = 1 given b = 1. So, if you consider the distinguishing advantage of the algorithm A which we have constructed, it is exactly the same with which the existing algorithm D can distinguish apart the experiment H0 versus experiment H1.

So, if the existing distinguisher can significantly distinguish apart the experiment H0 from experiment Hint, then what we have shown is an algorithm A which can significantly distinguish apart a pseudorandom sample generated by algorithm G from a uniformly random sample, but that is a contradiction to the assumption we are making, we are saying that the existing algorithm G is a secure PRG.

That means, since the existing algorithm G is a secure PRG, the distinguishing advantage of algorithm A is going to be upper bounded by a negligible probability, which further implies that the distinguishing advantage of the existing algorithm D is also going to be upper bounded by negligible probability. So, that proves our first claim.

(Refer Slide Time: 18:44)



In the same way, we can prove that if the existing algorithm G is secure, then no polynomial time distinguisher can significantly distinguish apart an instance of the experiment Hint from

the instance of the experiment H1. Again, the proof idea will remain the same. Assume for the moment you have an existing distinguisher who can distinguish apart the experiment Hint from H1 one. Using that, we design another distinguisher A, who can distinguish apart a pseudorandom sample of size big L bits from a uniformly random sample of size big L bits.

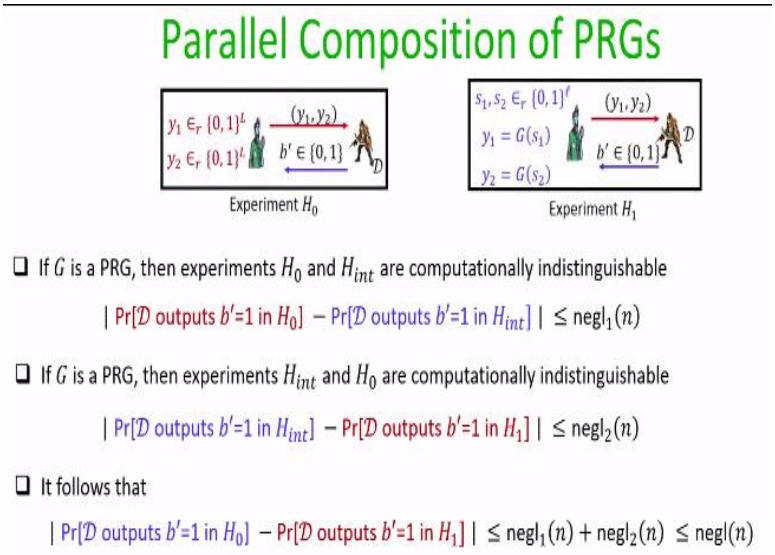
So, it participates in an instance of the indistinguishability based experiment, where it is given a sample y2 which is generated either uniformly randomly or it is generated by running an algorithm G with a uniformly random input. The goal of the adversary A is to find out whether b = 0 or b = 1. Now, what this A is going to do is it is going to pick a seed itself, which is of size little l bits, and it produces a pseudorandom sample y1 and it produces now a bigger challenge sample for the existing distinguisher D by concatenating the pseudorandom sample y1 with the challenge sample y2.

So before we proceed further, you can clearly see here that if the challenge sample y2 is uniformly random, then a pseudorandom sample y1 followed by a truly random sample would look like a sample which the D expects in an instance of the experiment Hint. Whereas if the sample y2 is pseudorandom sample, then a pseudorandom y1 concatenated with a pseudorandom y2 would create a sample for D as per an instance of the experiment H1, right.

So, based on the same idea which we use to prove the previous claim, we can actually end up showing that the probability with which A could distinguish apart whether the challenge y2 is pseudorandom or truly random is exactly the same with which the distinguisher D could distinguish apart whether it is participating in an instance of the experiment Hint versus whether it is participating in an instance of the experiment H1.

So, if the distinguishing advantage of algorithm G is non-negligible, then the distinguishing advantage of our algorithm A is also non-negligible, but that is a contradiction to the assumption that our algorithm G is pseudorandom.

(Refer Slide Time: 21:17)

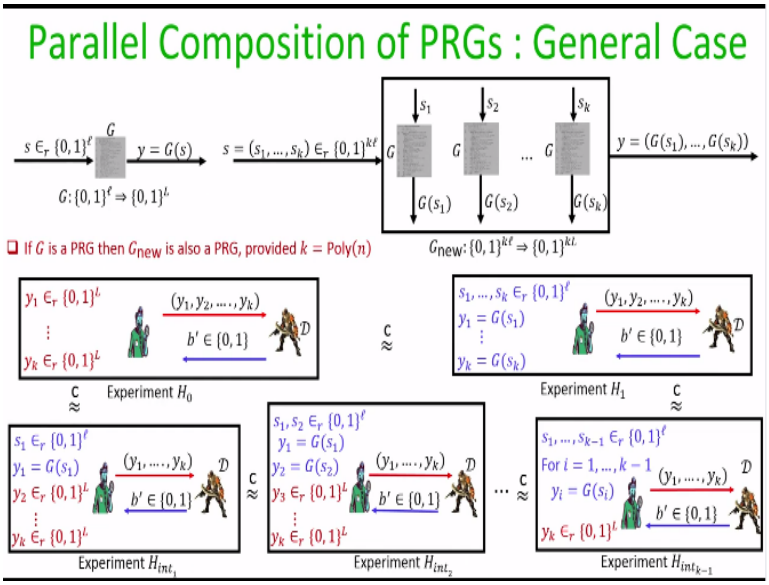


So, based on this, the summary of the proof is as follows. We have actually proved 2 individual claims. If the existing algorithm G is a pseudorandom generator, then no polynomial time distinguisher can distinguish apart whether it is participating in experiment H0 or whether it is participating in experiment Hint except with some negligible function say negligible 1.

In the same way if the existing algorithm is a secure PRG, then no distinguisher can distinguish apart whether it is participating in an instance of experiment Hint versus whether it is participating in an instance of experiment H1 except with a negligible probability which I denote by negligible 2. So if I sum these 2 distinguishing advantages, I get that the experiment H0 and experiment H1 are also computationally indistinguishable because the sum of 2 negligible functions is also upper bounded by a negligible function.

That means, if we actually compose the existing algorithm G twice with independent inputs, then the new algorithm is also a secure PRG.

(Refer Slide Time: 22:27)



So, let us come to the general case, right. The general case was when we were actually composing the existing algorithm G polynomial number of times, and to prove that the new algorithm is also a secure PRG. Namely, we create 2 instances of the experiment H0 and H1 where H0 would have actually created a challenge sample for the distinguisher generated as per running the truly random generator k independent times, whereas an experiment H1 the sample which is given to the distinguisher is actually generated by running the algorithm G k times independently.

Our goal is to prove that no polynomial time distinguisher can distinguish apart whether it is participating in experiment H0 or whether it is participating in experiment H1. To prove this claim basically, we have to now introduce polynomial number of intermediate hybrid experiments, right. Namely, we have to introduce k instances of intermediate hybrids. So, the first intermediate hybrid will be almost identical to H0 except that the first part of the challenge.

Namely the first block of the challenge sample which is given to the distinguisher is actually generated by running an instance of a pseudorandom generator, whereas the remaining blocks of the challenge sample which is given to the distinguisher are all generated uniformly random. So, that is the only difference between the experiment H0 and Hint1 and we can prove using similar strategy that we have used in the previous claim of the previous example that if the algorithm G is a secure PRG, then the distinguisher D cannot distinguish apart an instance of experiment H0versus an instance of experiment Hint1.

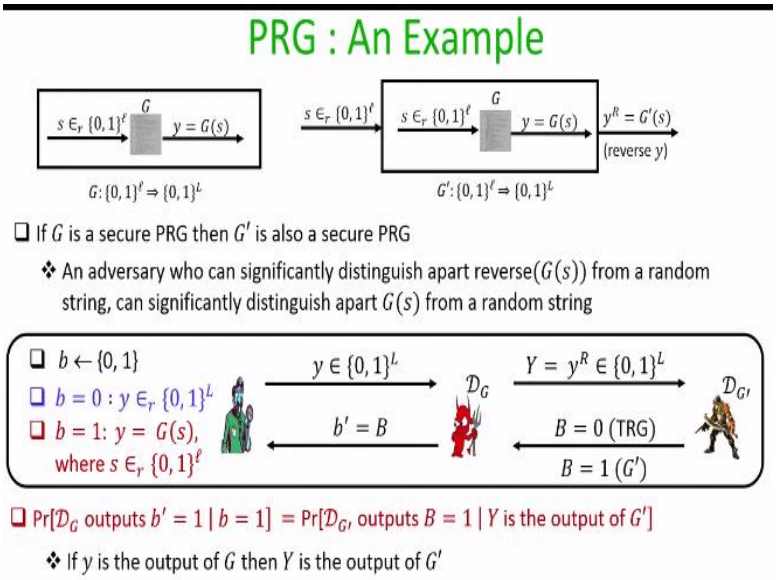
In the same way, the second intermediate hybrid experiment will be almost identical to the first hybrid experiment Hint1. The difference will be that the second part or second block of the challenge sample which is given to the distinguisher is now generated by running a pseudorandom generator and the remaining k minus 2 blocks of the challenge are generated uniformly random, right. Again, we can prove that if the existing algorithm G is a secure PRG, then no polynomial time distinguisher can distinguish apart an instance of experiment Hint1 from an instance of the experiment Hint2.

Like that, the k minus 1 at intermediate hybrid will be as follows. Here, the first k minus 1 blocks of the challenge sample which is given to the distinguisher is generated by running k minus 1 independent instances of the algorithm G and the last block of the challenge sample is generated uniformly random and we can prove that if my existing algorithm G is secure PRG, then no polynomial time distinguisher can distinguish apart an instance of this k minus 1th intermediate hybrid experiment from k minus 2th word intermediate hybrid experiment.

Finally, we will prove that if the existing algorithm G is secure, then no polynomial time distinguisher can distinguish apart the experiment H1 from an instance of the k minus 1th intermediate hybrid experiment. So, if I now sum up this k distinguishing advantages of the adversary, what I end up showing is that the distinguishing advantage of any polynomial time distinguisher to distinguish apart an instance of the experiment H0 from an instance of the experiment H1 is upper bounded by some k times negligible function.

Since k is a polynomial function, k times negligible function is also going to be a negligible function, which proves that my algorithm G new is also a secure PRG.

(Refer Slide Time: 26:18)



So now, let us look into the final example for this lecture. We are now considering some other operation which we can perform on PRG and obtain secure PRG. So, here I am given some arbitrary secure PRG and I am now constructing a new PRG G dash where the output of G dash is simply obtained by running the algorithm G, which is the existing algorithm and by simply reversing the output of the existing algorithm G, right. My claim is that if your existing algorithm G is a secure PRG, then this new algorithm G dash is also a secure PRG.

Again the proof will be by reduction and intuition behind the reduction group is as follows. On contrary, assume that your new algorithm G dash is not a secure algorithm. That means, assume there exist an algorithm polynomial time distinguisher who can distinguish apart the output of G dash from an outcome of a truly random generator, then using that algorithm we can also actually design a polynomial time distinguisher who can distinguish apart an outcome of an algorithm G from an outcome of a truly random generator, which will be a contradiction.

The intuition behind this reduction is that reverse of any uniformly random string is also a uniformly random string and the reverse of a pseudorandom string is also a pseudorandom string. Namely, the idea behind a reduction is as follows. So, assume for instance, you have an existing distinguisher the DG dash for the new algorithm for the G dash we have constructed and using this algorithm I want to construct another polynomial time distinguisher DG for my existing algorithm G. So the distinguisher DG is given a sample, which is either generated uniformly randomly or by running the algorithm G.

What this algorithm DG is going to do is it is going to simply produce a new sample for my algorithm DG dash by simply reversing the bits of the challenge sample y. So, before proceeding further in the reduction, the point here is that if the sample little y is actually a uniformly random sample, then so is the new sample big Y. On the other hand, if the sample little y is a pseudorandom sample, then so is the new sample big Y.

That means with whatever probability my existing algorithm DG dash can distinguish apart a truly random sample big Y from a pseudorandom sample big Y, with almost the same probability, my new algorithm DG is going to distinguish apart a uniformly random sample little y from a uniformly random sample big Y, that is basically the idea behind a reduction, and I am leaving the full details of the reduction for you. Basically, we end up showing the distinguishing advantage of my algorithm DG is exactly the same as the distinguishing advantage of the existing algorithm DG dash, right.

So, if that prove if my existing algorithm G is a secure PRG, then so is the new algorithm G dash. So, that brings me to the end of this lecture. Just to summarize, in this lecture, we have seen a new primitive called pseudorandom generator, which is a deterministic algorithm and the goal of the pseudorandom generator is to expand its input and generate an output which is significantly larger than its input. More importantly, the goal of the pseudorandom generator is to produce an output sample which looks almost identical to an output which would have been generated by a truly random generator.

We have seen various equivalent definitions of pseudorandom generator, and we have also seen how we can parallely compose pseudorandom generators polynomial number of time to obtain a new pseudorandom generator. I hope you enjoyed this lecture. Thank you.