CSA E0 235: Cryptography 28 Aug 2019

Scribe for Lecture 7

Instructor: Arpita Patra *Submitted by: Sqn Ldr Jitender Jamwal*

1 Recall

In the previous lecture we have learnt about proving the fact that *Pseudo Random Generator (PRG)* with one bit expansion implies *PRG* with polynomial expansion, and for proving this statement we introduced *hybrid arguments.* We also listed some applications of *PRG* namely commitment schemes and further we used these to construct a coin tossing problem over telephone. We saw that in the *hybrid arguments* the idea is to divide the bigger problem into few smaller ones, then solve the smaller problems and combine the solutions to get the final statement or final theorem which we want to prove. In the only example we had worked out with the hybrid proofs, the smaller problems that were defined from the bigger problem were of the same class. But it may not be necessarily the case always. But for this course, the hybrid arguments we are going to learn will be of this type only. The subproblems would almost be identical to the original problem.

2 Today's Goal

In this lecture we will discuss about the following :

- The first goal is to introduce a stronger class of attack which is termed as **Chosen Plaintext Attack (CPA).** This attack will only be applicable to encryption schemes in which the keys are reused. If the keys are not reused, then this attack reduces to the previous form of attack i.e. **Ciphertext Only Attack (COA).** So, CPA works for schemes in which the same key is used for encrypting multiple messages.
- The next goal would be to relate it to other close security notions like *mult-coa, mult-cpa* etc.
- Thereafter the next goal would be to realise this new security definition.
- Our next task will be to find out assumptions to build a construction for CPA. The notion which we are going to need to build a construction for CPA security is called **Pseudo Random Function (PRF)**. This is a more complicated notion as compared to the previously seen notion of *PRG.*
- And finally, we will work out the proof.

3 Security for SKE with COA to CPA

So far we have captured *Ciphertext Only Attack*. Now we will be going to capture *Chosen Plaintext Attack*. The security challenge remains same, i.e., given the knowledge of two messages (or vector of messages), it cannot be distinguished whether the ciphertext corresponds to the first or second message (or message vector). In CPA, we assume the distinguisher to be randomised and Probabilistic Polynomial Time (PPT), as it was assumed in the case of COA.

4 Chosen Plaintext Attack : Introduction

In *Chosen Plaintext Attack*, as the name suggests, the adversary influences the sender to encrypt the messages of his choice. So basically, the adversary picks up the message, which he wants to be encrypted, and gets the corresponding ciphertext, and this can continue as many times as adversary wants. The adversary is adaptive, in the sense that he chooses the first message and gets the corresponding ciphertext. After looking at the ciphertext, he can use the knowledge gained by it and then he picks up second message of his choice which may be different from the first message.

Once the adversary has the set of pairs of all messages of his choice and their corresponding ciphertexts, which can be referred to training ciphertexts, he sets his goal to determine the plaintext encrypted in a new challenge ciphertext. It is to be noted that the entire training ciphertext and the new challenge ciphertext has been prepared by using the same key. In case a new key is used every time to encrypt every new message then there is no security problem. But this is not desired in any secret key encryption scheme. Ideally, in any secret key encryption scheme, we want to reuse the keys. Therefore, CPA attacks are feasible for any secret key encryption scheme.

CPA attacks are stronger because after receiving the challenge ciphertext, the adversary can again acquire the training ciphertexts of the messages of his choice. We can say that the adversary has got an encryption *Oracle* which can be considered as a black box in which there is a key hidden inside and that can output the ciphertext for any given input plaintext.

5 Is CPA Realistic?

An obvious question which arises in our mind is that, is this type of attack really *mountable* in practice? Why should the sender be interacting with the adversary and providing the training ciphertext for the messages of his choice? But practically CPA is mountable and has happened in the past history. In fact CPA was the reason which shortened the second world war by 2 to 3 years.

During the second world war, Germany was using encryption scheme for communication and Britain was trying to break it. Britain was able to see a lot of ciphertext of Germany by eavesdropping and planned to get the ciphertext for the messages of their choice. So, British army put some sea mines in the British Channel at predetermined locations and they believed that the Germans will come and will send back the location information to

their headquarters using their encryption scheme. The same exactly happened. Germany discovered the locations of sea mines, encrypted the location messages and sent back those messages to their headquarters. These encrypted messages were eavesdropped by the British Army. This ciphertext was given to the father of computer science, Alan Turing, to break the encryption scheme. This scheme is exactly Chosen Plaintext Attack because the locations were decided by British Army which means that the messages of the choice of British Army were encrypted by the Germans, and the corresponding ciphertext was eavesdropped.

There are some other scenarios in which the algorithms of encryption schemes and the secret key are hardcoded in a secure hardware. Such hardware is used in many military applications. An insider may have access to such hardware and then he can choose messages of his choice and break the encryption scheme.

So, as constructive cryptographers our goal is to design encryptions schemes keeping such type of attacks in mind and make those schemes prune to such attacks.

6 CPA Indistinguishability Experiment

So now we introduce a *game* to capture the CPA attack (Figure 1). The game starts with a *training phase* in which the attacker is given the *Oracle* access to the encryption scheme. The *attacker* queries the *challenger* with the plain text. The *challenger* picks up a key once for all for the game and then does not change this key thereafter. By using that key, the *challenger* prepares the corresponding ciphertext and responds to the *attacker*. This phase is the pre challenge training phase and goes on as many times as *attacker* wants.

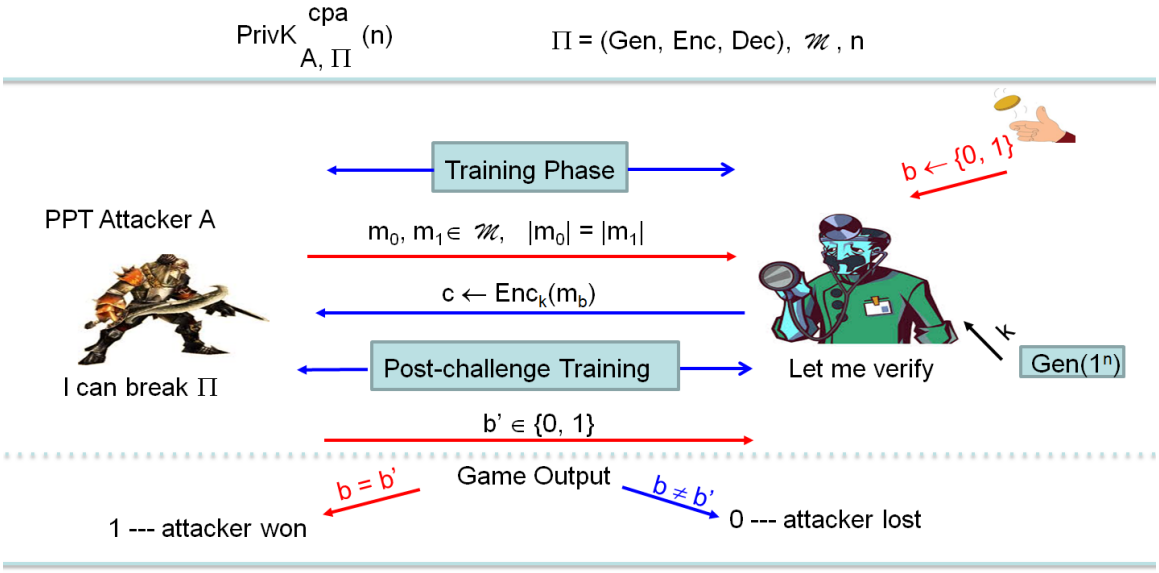

Figure 1: CPA Indistinguishability Experiment

Thereafter, comes the *challenge phase* in which the *attacker* submits two messages. These two messages can include the ones which were already queried during the training phase. The *challenger* picks a uniform bit b and encrypts the message number corresponding to the flip bit, i.e. m_0 or m_1, and submits the ciphertext to the attacker.

After looking at the ciphertext, the *attacker* is again given an opportunity to train himself. This phase is known as post challenge training phase. The attacker chooses messages of his choice and is responded back with the ciphertext corresponding to those messages using the same key.

Finally comes the *response phase*. The *attacker* is provided with adequate training and now he has to guess which of the two messages (m_0 or m_1) was encrypted. The *attacker* responds with the bit b'. And as usual, the output of the game will be 1 if the *attacker* wins ($b' = b$) or zero if *attacker* loses ($b' \neq b$). The scheme is said to be CPA secure if for every Probabalistic Polynomial Time attacker, there is a negligible function *negl*, such that :

$$\Pr\left(\text{PrivK}^{cpa}_{A,\pi}(\text{n}) = 1\right) \leq \tfrac{1}{2} + \text{negl(n)}$$

7 Relation with other close securiity notions

Now we will describe the relation of CPA with other security notions and we will do it for two closely related security notions.

7.1 Mult-COA

The first notion is *mult-COA* where in the two challenge messages are not single messages but they are a vector of messages. Now, suppose both the vector messages are of length **L**, then by choosing **L -1** messages, which are same in both the vectors, the *attacker* knows that the *challenger* will encrypt the same messages whether he chooses the first vector or the second vector, because the same set of messages are there in both the vectors. So this can act as a training phase.

The challenge messages are picked up as m_i and m_j, which are different, and are in the middle of both the vectors. So the left and the right messages in both the vectors are same. This can emulate the pre-challenge and post-challenge training phase respectively.

If we compare this scheme with CPA then we can analyse that this is not adaptive, because in this scheme all the messages are sent at once in the form of a vector, rather than analysing the ciphertext of each message and then making a strategy to choose the next message. Therefore, *mult-COA* is weaker as compared to CPA. There are schemes that are *mult-COA* secure but not CPA Secure. All schemes which are CPA secure are also *mult-COA* secure.

7.2 Mult-CPA

The other security notion which is related to CPA is *mult-CPA*. In this scheme, during the challenge phase, rather than a pair of two messages, a pair of two message vectors is sent by the *attacker*. CPA and *mult-CPA* are equivalent and *hybrid arguments* technique can be applied to prove this equivalence relation.

The two extremes can be considered as follows. The first extreme can be the ciphertexts corresponding to all the messages of the first vector and the second extreme can be the ciphertexts corresponding to all the messages of the second vector in the challenge phase.

The intermediate views can be considered as the following. In the first intermediate level, the first ciphertext corresponding to the first message of the second vector and rest corresponding to all the remaining messages of the first vector. In the second intermediate level the first two ciphertexts corresponding to first two messages of the second vector and remaining corresponding to the remaining messages of the first vector and so on. Then by taking any two consecutive hybrids, the *mult-CPA* can be reduced to single CPA scheme.

8 Assumptions of CPA-Secure Scheme

Now we move to search for assumptions of CPA secure scheme. The first thing which we can recall is that to employ an encryption scheme we must use randomisation. The schemes which do not employ randomisation can be broken with *mult-COA* attack. As CPA is stronger then *mult-COA,* therefore randomisation is needed in CPA also. Since the encryption procedure will be randomised we can expect that for the same message we can get different ciphertexts. This justifies the liberty given to the *attacker* to choose the same set of messages during the training phase. But at the same time a single key is to be used. One alternative can be to use a key which has very large length. The second alternative is that the two parties agree on a key which is a function. The function has an associated truth table as shown in figure 2. The function has a special property that it is picked uniformly at random from a space of all functions which have domain and codomain as specified in the truth table.

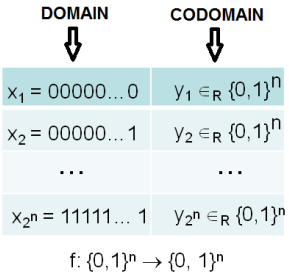

Figure 2: Truth Table of a function

Let us denote the domain and co-domain of the function by x_i and y_i where each x_i and y_i are strings of size n bits. This function is known as **Truly Random Function (TRF)**. Whenever a message is to be encrypted, an x_i is picked up uniformly at random from the domain. Once an x_i is picked up then we go to the corresponding y_i which will be used as one time pad for the message. The information about the random x_i is also passed to the receiver. So the ciphertext will contain information about the x_i concatenated with the message XORed with y_i.

The receiver also has the same truth table. On receiving the ciphertext the receiver goes to the corresponding x_i row in the truth table, gets the corresponding y_i and decrypts the message. It is equivalent to many instances of one time pad with almost different keys each time. But the problem with the above solution is that we are not following the notion of using a single small key. The size of the key in the above solution is equal to the size of the truth table which is $n2^n$. So in our scheme we will replace this huge truth table with something which is more elegant but emulate the randomisation of this huge truth table. The solution to this problem is to use **Pseudorandom Functions (PRFs)** which do not have the size of TRF but emulate its randomness. This concept of PRFs was brought in 1986.

9 Pseudorandomness

Lets get an insight about *pseudorandomness*. It is a property of a probability distribution on a set. In case of PRG this set was a simpler set of strings. But in case of PRF, this is a set of all functions and is much more complex in nature. Let G be the probability distribution on the set of all functions. We can say that G is pseudorandom if a function drawn according to G is indistinguishable from a function drawn according to a uniform probability distribution on the same set by a distinguisher who is allowed to make polynomial number of queries on the set. For this purpose we cannot give the entire truth table of the functions to the distinguisher, rather we provide an *Oracle* access to him. By *Oracle* access we mean that for any number of queries for x by the distinguisher, he is provided with the corresponding y. Inside the *Oracle* box we can consider a sampler which will choose a function either according to probability distribution G or uniform probability distribution U. And this function is chosen once for all. A function which is drawn according to U is called a truly random function and a function which is drawn according to G needs certain extra properties to be called as a pseudorandom function.

There are two ways to count the size of such functions. One way to count the number of functions is as we can see that there are total 2^n x values and each x has 2^n possibilities. The other way is to consider the entire truth table as a huge string and the length of such string would be $n2^n$. Thus, the number of functions would be 2^{n2^n}. Now if we consider the uniform probability distribution in this huge space then each function in the set will have probability of occurring $1/\ 2^{n2^n}$. A truly random function is chosen according to the uniform probability distribution and it has a property that its output behaviour is completely unpredictable. By this we mean that if we take any x the corresponding y will be a uniform string.

10 Pseudorandom Functions

A pseudorandom function is chosen from the space of all possible functions in *domain* and *codomain*. It has the following properties :

- It is chosen according to some distribution function which is different from uniform probability distribution function, yet its output behaviour must look like a TRF, that is the output must look like a pseudorandom string.

- It must be efficiently computable, that means given a value of x, the corresponding value of y should be computed quickly.
- It should be precisely representable and must not be represented by a truth table which has a huge size of $n2^n$.

The specifications of the function will be shared by the sender and receiver and will be the key. So pseudorandom functions are the *keyed* functions. The set of *keyed* functions is a small subset of the total set of functions and has size 2^n. A pseudorandom function will be a function chosen uniformly at random from the set of *keyed* functions and behaves like a truly random function. We allow the polynomially bounded distinguisher to talk to the chosen function as many times as he wants, rather than providing a huge set of functions to the distinguisher.

11 Indistinguishability Game for PRF

Let us discuss the Indistinguishability game for PRF. On one hand, we have the set of all functions with uniform probability distribution over it. Function chosen from it will be known as truly random function. On the other hand, we have a much smaller space that is the set of all *keyed* functions. The function chosen from this set of functions will be known as pseudorandom function. The *distinguisher* and the *challenger* will be playing the following game.

In the beginning, the *challenger* decides uniformly at random whether he is going to pick up a TRF or a PRF. This is a one time process and the function is not changed for each and every query. If the value of flip bit b is zero then the *challenger* picks up a truly random function and all the queries of the *distinguisher* are answered according to TRF. Thus, for any value of x, the *challenger* will return a random y value. It is to be noted that these queries are *adaptive*. On the other hand, if the value of flip bit b is one, then the *challenger* picks up a pseudorandom function, which means he picks up a key K uniformly at random from the key space which is the set of all strings of length n and then he will reply all the queries of the *distinguisher* with respect to this particular function. Mathematically, a function F chosen according to the scheme discussed above is said to be a pseudorandom function if for every probabalistic polynomial time *distinguisher* D, there is a negligible value $negl(n)$ such that :

$$\left| Pr[D^{F_{\mathrm{k}}(\cdot)}(1^n) = 1] - Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq negl(n)$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $f \in Func_n$ and the randomness of D. The symbol $(\cdot)$ in the above equation means that the distinguisher is provided with *Oracle* access.

12 Pseudorandom Function: Construction

Lets work out an example to construct a PRF and then we will show that the constructed function, although having all the required properties, will not be a good pseudorandom function. Consider a length preserving function F which is a *keyed* function with the property that if the input is x, then the output y is $k \oplus x$, where k is the key. In order to break this scheme, we can provide input x as zero which will give key k as output. But if a pseudorandom function has been picked up, then it is not guaranteed that the key k is indeed the original key.

A valid strategy would be to send one more query. Let the two queries be x_1 and x_2. These queries will be answered according to a truly random function if the flip bit is zero or will be answered according to a pseudorandom function if the flip bit is one. By looking at the structure of the function, it is very easy for the *distinguisher* to check whether he is talking to a truly random function or to a pseudorandom function. The *distinguisher* will simply XOR the two messages and if $x_1 \oplus x_2 = y_1 \oplus y_2$, then the *distinguisher* outputs bit one which indicates that he is talking to a pseudorandom function. Otherwise if $x_1 \oplus x_2 \neq y_1 \oplus y_2$, then the *distinguisher* outputs bit zero which indicates that he is talking to a truly random function. The probability that the *distinguisher* knows that he is interacting with the *keyed* function is 1. And the probability that the *distinguisher* will output 1 even if he interacts with the TRF is 2^{-n}. If we look at the probability difference between these two, it will be 1 - 2^{-n}, which is definitely not negligible and thus the constructed function is not a PRF.

13 Do PRFs exist?

So now comes the obvious question that whether PRFs exist or not? There is no proof for this but it is strongly believed that they exist. So this will be yet another assumption for this course that PRFs do exist. Later in the course we will see that PRGs imply PRFs. There is a famous Goldreich-Goldwasser-Micali construction which apply *hybrid arguments* techniques to formulate PRFs from PRGs. On the other hand, in the practical front, we have constructions for PRFs which are extremely efficient and the reason that we believe they are PRFs is that we don't have good distinguishers for them so far. These constructions are called as block ciphers like DES and AES (corresponding versions of PRGs are known as stream ciphers). To be more specific, DES, AES and 3DES are **Pseudo Random Permutations (PRPs)**. PRPs are related to **Truly Random Permutations (TRPs)** in the same way as the pseudorandom functions are related to truly random functions. There are various such notions which can be represented in a petal form as shown in figure 3 and are known as sibling notions.

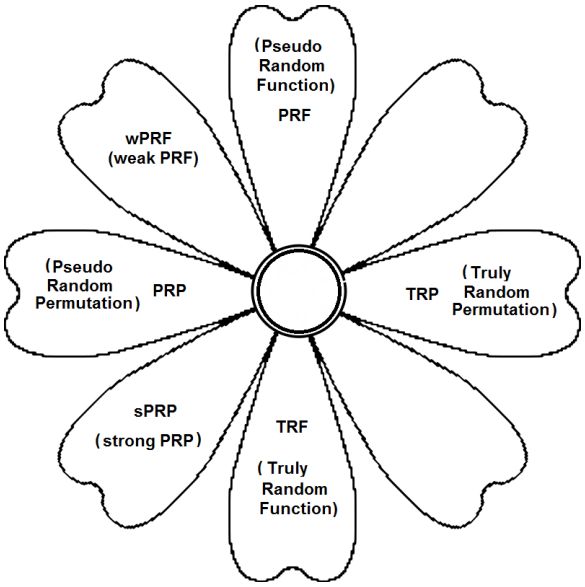

Figure 3: Sibling Notions

14 References

[1] Arpita Patra, `https://www.csa.iisc.ac.in/~cris/e0_235.html`. Course Materials.

[2] Jonathan Katz and Yehuda Lindell, *Introduction to the Modern Cryptography*

[3] Internet, *https://www.coursera.org/learn*