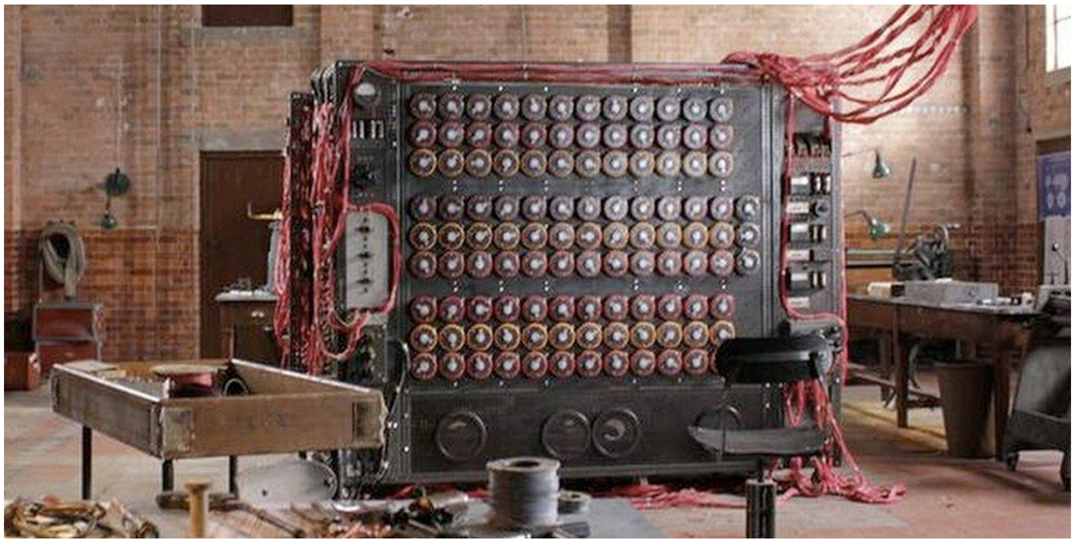
Application of Number Theory in Cryptography

Mentee : Soham Shrikant Joshi
210051004@iitb.ac.in

Mentor : Tirthankar Mazumder
20b090012@iitb.ac.in

July 24, 2022



Contents

1 Overview

The main objectives in this study are :

1. Learning how crypto primitives work
2. How to use them correctly and reason about security

1.1 Examples

1. Secure Communication (HTTPS)
2. Encrypting files on disk
3. Content Protection (CSS, AACS)
4. User Communication

1.2 Parts of security

1. Handshake Protocol (Establish shared secret key using public key cryptography)
2. Transmit data using the shared secret key

1.3 Secure communication

Secure communication basically deals with communication from one party at a point of time to another party at another point of time. A few examples can be :

1. Communication between a client and a server
2. Protected files on a disk. (This is basically communication between a person with another person in the future!)

2 Basic Building Blocks

The basic building block of secure communication is *Symmetric Encryption*. In this method a shared key is established between the two parties (The key is established magically for now, the process will be elaborated in a later report). After this, the same key is used to encrypt as well as decrypt the message where the encryption and decryption is done using *ciphers*.

A Cipher is basically a function from a string of characters to another string of characters which takes a key as an input parameter! Hence, an encryption cipher is a mapping from plaintext space (normal messages) to ciphertext space (encrypted messages) and a decryption cipher is a mapping from ciphertext space to plaintext space.
The schematic of symmetric enryption and decryption is shown below :

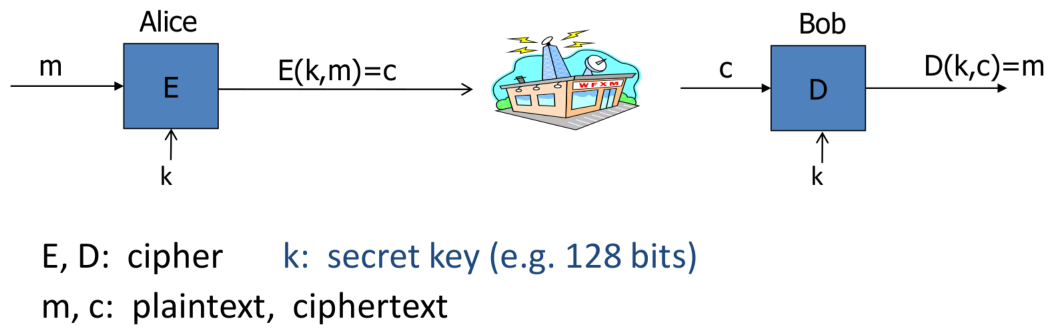


Figure 1: Symmetric Encryption

2.1 Keys

The keys I mentioned earlier are of two types with respect to use cases :

1. Single use key
   (a) Key is used to encrypt one message
   (b) Example : Encrypted email (New key generated for every email
2. Multi-use key
   (a) Key used to encrypt multiple messages

   (b) Needs more machinery than for one-time key
   (c) Example : Encrypted Files (Same key used for multiple files)

2.2 Utility of Cryptography

We are often under the impression that the systems we use are "perfectly" secure due to cryptography. However, that impression is not correct.

1. Cryptography is :
   (a) A tremendous tool
   (b) The basis for many security mechanisms
2. Cryptography is NOT :
   (a) The solution to all security problems. (example : software bugs, social attacks)
   (b) Reliable unless implemented and used properly
   (c) Something which you should try to invent yourself (there have been many many examples of broken ad-hoc designs)

2.3 Secure Multi-Party Computation

Let us say we have an election with 10 candidates and 100 voters. Normally, the 100 voters each go to a voting station, cast their vote and the votes are counted by a "trusted authority". This approach has a big issue, the authority might not be trustworthy in the first place ! In order to solve this, a theorem of cryptography comes to our rescue.

2.3.1 Decentralisation Theorem

This theorem states that,
Anything that can be done with a trusted authority can also be done without.

That is, Our problem can be solved using decentralisation! The parties can simply talk amongst themselves and there is a function that will convert the information obtained to the winner of the election.

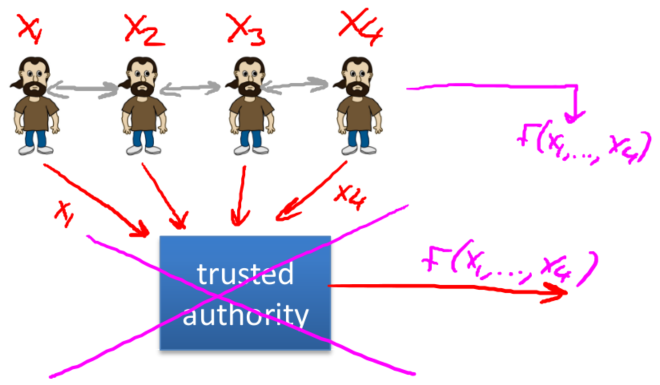


Figure 2: Decentralisation

2.3.2 Zero Knowledge

This can only be described as "magic". A person (Alice) can convince Bob that she knows something without actually giving up that piece of information. This is essentially the gist of a zero knowledge proof.
For instance if there is a number N then Alice can convey that she knows the factors of that number N without actually disclosing the factors (the specifics of how this works will not be covered here).
This is a particularly useful method of communication even from a cryptographic standpoint because we can convince that the winner in our election example (Figure 2) is genuine using a zero knowledge proof, hence, not disclosing any information about the actual votes in case the results are challenged!

2.4 A rigorous science

Cryptography is indeed a rigorous science. Any cryptographic setup consists of the following steps :

1. Precisely specify threat model
2. Propose a construction
3. Prove that breaking construction under the threat model will solve an underlying "hard" problem.

The meaning of these points will become clear as we take examples of cryptographic constructions in this report.

3 Historic Ciphers

The schematic of a symmetric cipher has been covered earlier (Figure 1). However, to understand this and truly appreciate the methods of cryptography let's start one of the most interesting place to be, the beginning.

3.1 Caesar's Cipher

It is unknown how effective the Caesar cipher was at the time, but it is likely to have been reasonably secure, not least because most of Caesar's enemies would have been illiterate and others would have assumed that the messages were written in an unknown foreign language.

The Caesar's Cipher works by shifting each letter of the english alphabet forwards by 3 places.

For example :

1. a maps to d
2. b maps to e
3. c maps to f

and so on till z maps to c

3.2 Substitution Cipher

The substitution cipher works similarly to a Caesar's cipher, just the map is not restricted to a shift by 3 places.

For instance,

1. a - c
2. b - w

3. z - a

and so on is a valid substitution cipher.

3.3 Breaking the Substitution Cipher

The substitution cipher at first glance, looks very secure. But as it turns out, it has been broken! (and not just broken, it is badly broken). Before any plan of attack on this type of encryption, we must first gauge the opponent's resources, that is, the possibilities available.

The english language has 26 letters (spaces remain unaffected). And the mapping of a substitution cipher has to be one-one and onto (since, on reading the ciphertext, a person with the "table" of mappings should be able to figure out the message). Hence,

$$sizeofkeyspace(S) = 26! \approx 2^{88} \tag{1}$$

This is a very decent size of key space. Even by modern standards, iterating through all elements of the key space to get a meaningful message is not feasible. Hence, the size of key space cannot be our point of attack. However, this system has a fatal flaw . All letters do not have the same frequency of occurence in the English language! For instance, some frequencies are given below

1. "e" : 12.7%
2. "t" : 9.1%
3. "a" : 8.1%

Hence, the letter "e" is the most common letter of the english language. So it is natural to guess that the most occuring letter in the ciphertext is the map of e under the substitution cipher. Moreover, we can compare the statistical occurences of pairs of letters and so on to guess the mapping and hence, obtain the plaintext!

Thus, the substitution ciphers fails even under a "ciphertext only attack" (the intruder doesn't even try to attack the key!). To give an idea of how bad this encryption is, we might as well send plaintext over the network and it won't make much of a difference.

3.4 Vigenere Cipher

This cipher had originated in the renaissance period and uses an interesting mechanism to encrypt things. It uses a "key" which is an n-lettered word (e.g. crypto) and copies this word till the length of the copied key equals the plaintext (e.g. cryptocryptocryp). Now we take the addition modulo 26 of the plaintext and the copied key.

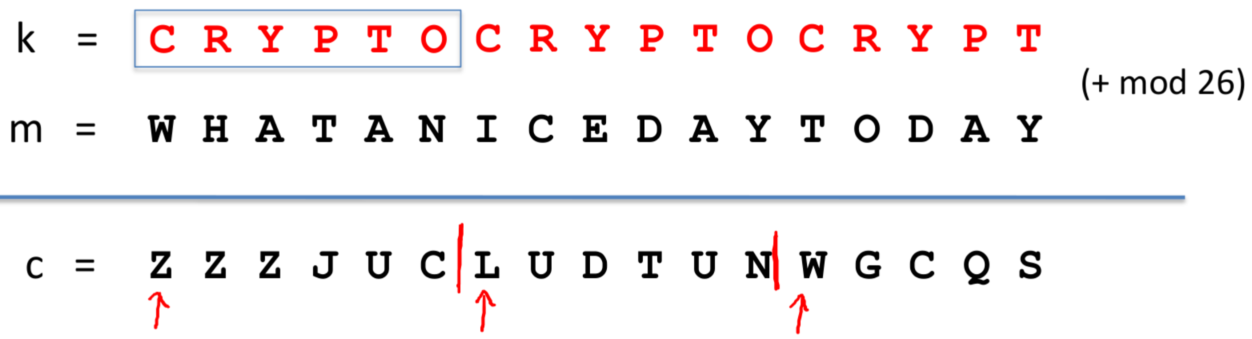


Figure 3: Vigenere Cipher

For all practical purposes we can have the following threat model :

1. The adversary knows the length of the key.
2. The adversary knows the ciphertext.

Adversary won't actually know the length of the key beforehand but it is fairly easy to iterate through all lengths till a sensible message appears. So assumption 1 makes sense. Now, we proceed to breaking the vigenere cipher.

3.5 Breaking the Vigenere Cipher

Let's continue with our example (Figure 3), and say the adversary is aware that the length of the key is 6. We can again use the same logic as the substitution cipher to break this cipher. Let us examine the first letter and every 6th letter after that. These letters have been encrypted using the same letter (since the size of the key is 6). Now, we can conduct a frequency analysis on the distribution of these letters in the ciphertext and we know the letter that "e"(or some high frequency letter) is mapped to ! We can repeat this analysis on 2nd, 3rd, 4th, 5th, 6th elements to recover the key

and hence, the plaintext !
Note that, the size of the key space is of the order 26^k . Hence, brute forcing won't work here either.

3.6 Rotor Machines

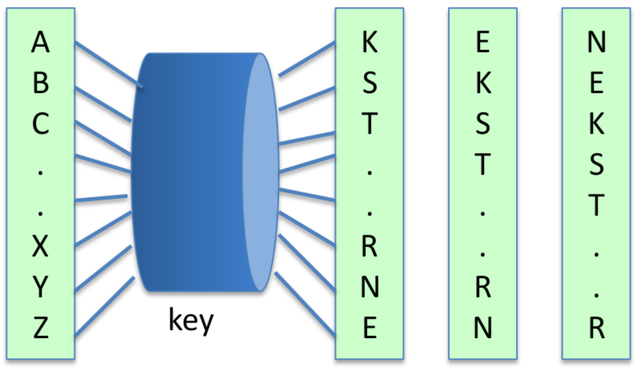


Figure 4: Rotor Machine

A rotor machine is kind of an extension of the substitution cipher. After every keypress, the rotor rotates by 1 unit, causing the key to shift by one unit creating a new key. The most famous example of a rotor machine is of course the enigma.

3.6.1 The Enigma

The Enigma machine is a cipher device developed and used in the early- to mid-20th century to protect commercial, diplomatic, and military communication. It was employed extensively by Nazi Germany during World War II, in all branches of the German military. The Enigma machine was considered so secure that it was used to encipher the most top-secret messages.

The enigma machine had 4 rotors whose starting combination was sent at the beginning of every day.

Ways of starting configurations = $26^4 \approx 2^{18}$ ways

If the optional plugboard is included, the keyspace increases to a size of 2^{18} keys.
By Modern Standards, a keyspace of size 2^{18} is nothing, can be bruteforced

Figure 5: The Enigma Machine

in a matter of few seconds. Alan Turing was eventually able to crack the enigma and alter the course of the second world war !

3.7 Data Encryption Standard (DES)

Around 1974, the government wanted a decent cipher for encrypting stuff, so IBM put together the DES as a standard for encryption. This has the following properties.
keys = 2^{56} block size = 64 bytes
Due to a block size of 64 bytes, 8 characters are encrypted at a time. By modern standards the key space is a bit too small and can be broken by brute force search. Hence, this is not used in current times.

3.8 Modern Encryption

Now that we have taken a look at some ciphers, let's take a look at some ciphers that are actually considered secure :

1. AES (2001)
2. salsa20 (2008)

Modern ciphers typically use a 128-bit key. We might encounter some of these ciphers later in the report.

4 Stream Ciphers

This section assumes that you have some knowledge of discrete probability. If not, you can try to follow along and google the parts that seem a bit fuzzy. A symmetric cipher is defined as follows :

A cipher defined over (K, M, C) is a pair of "efficient" algorithms (E,D) where E : K $\times M \to C, D : K \times C \to M, : \forall m \in M, k \in K : D(K, E(k, m)) = m$

1. E is often randomized.
2. D is always deterministic

I have used the following notation in the above definition :

1. K = set of all possible keys
2. M = set of all possible messages/plaintexts
3. C = set of all possible ciphers

Here, by efficient algorithms, we can mean two things. From a theoretical perspective, the program runs in a polynomial time, from a practical perspective, the cipher runs in some specified time limit (for example 100 seconds).

We have already seen some ciphers in section 3. However up until the 20th century all the ciphers used have been insecure. It seems that it is time for a bell labs engineer to make his mark on history.

4.1 The One-Time Pad (OTP)

The OTP was created by Vernam (a bell labs engineer) in 1917. This is the first example of a "secure" cipher. The OTP has

$$M = C = \{0, 1\}^n \tag{2}$$
$$K = \{0, 1\}^n \tag{3}$$

where $\{0, 1\}^n$ represents a string of n bits showing either 0 or 1. Thus, here we have a key which is as long as the message. In addition to this, vernam defined both the encryption and decryption to be an XOR operation. (XOR is denoted by $\oplus$)

$$
\begin{aligned}
c := E(k,m) &= k \oplus m && (4)\\
D(k,c) &= k \oplus c && (5)\\
D(k,E(k,m)) = D(k,k \oplus m) &= k \oplus (k \oplus m) = m && (6)
\end{aligned}
$$

Hence, the one time pad is a symmetric cipher. Interestingly, given the plaintext and the ciphertext we can compute the key using

$$
\begin{aligned}
c &= k \oplus m && (7)\\
k &= k \oplus m \oplus m = c \oplus m && (8)\\
k &= m \oplus c && (9)
\end{aligned}
$$

Well, this seems like a good cipher, but so did all the ciphers we saw before! So how are we sure that this cipher will withstand all attacks? To answer this question we need some theory which deals with some notion of how much “information” is this setup “leaking” via the ciphertext. It is time to meet another Bell Labs engineer, and not just any engineer, the father of Information Theory himself !

4.2 Information Theoretic Security

Shannon had already spent considerable time in Information Theory (1949). So it is only natural for the father of information theory to work in this domain. The basic idea in information theoretic security is that the ciphertext should not reveal any information about plaintext. Shannon defined perfect secrecy as follows :

A cipher (E, D) over (K, M, C) has perfect secrecy if $\forall m_0, m_1 \in M(|m_0| = |m_1|)$ and $\forall c \in C$

P[E(k,m_0) = c] = P[E(k,m_1) = c] where k $\overset{R}{\leftarrow} K$

that is, k is a uniformly random variable belonging to K.
Now that we have a definition of perfect secrecy, we have an important lemma to prove.

Lemma : OTP has perfect secrecy.

Proof of the OTP

Thus, OTP cannot have a ciphertext only attack !! However, we have some bad news.

Theorem : perfect secrecy $\Rightarrow |K| > |M|$.

However, this creates an issue. Since, we have a mechanism to exchange keys of length K we might as well use that same mechanism to the exchange the messages instead of doing this stuff! So we need a new definition of secrecy, one that can be applied in practical cases.

4.3 Making OTP Practical

The key idea here is replacing the random key (k) by "pseudorandom" key. Pseudo-random Generators(PRG) are defined as follows :

PRG : is a function G : $\{0, 1\}^s \rightarrow$\{0,1\}^{n}, n $\gg$ s, which is "efficiently" computable by deterministic algorithm.

$$c = E(k, m) := m \oplus G(k) \tag{10}$$
$$D(k, c) := c \oplus G(k) \tag{11}$$

where G(k) acts as a key of length same as the message. However, this creates an issue. the length of the actual key (k) is less than that of the message. Thus, by Theorem undeer section 4.2, Stream Ciphers (where key is generated using PRGs) cannot be perfectly secure! Hence, we need another definition of secrecy. This definition has to depend on the nature of the PRG as well.

So in a way, the security of the encryption depends on how "unpredictable" the PRG is. Let us go into this in a bit more detail.

4.3.1 Predictibility of a PRG

We say that G : K $\rightarrow$\{0,1\}^{n} is predictable if $\exists$ efficient algorithm A and $\exists 1 \leq i \leq n$ s.t

$$Pr[A(G(k))|_{1,\cdots,i} = G(k)|_{i+1}] \geq \frac{1}{2} + \epsilon \tag{12}$$

where k $\overset{R}{\leftarrow}$ K (k is uniformly random variable taking values in K). For some non-negative ϵ (say $\epsilon \geq 1/2^{30}$)

A PRG is called "unpredictable" if it is not predictable. $\Rightarrow \forall i$: no efficient adversary can predict the $(i+1)^{th}$ bit for "non-negligible" ϵ

For example, XOR(G(k)) = 1 is a predictable PRG. If we have k-1 bits, we can predict the kth bit.

4.3.2 Negligible and Non-negligible ϵ

1. In practice : ϵ is a scalar and
   (a) ϵ is non-neg. if $\epsilon \geq \frac{1}{2^{30}}$
   (b) ϵ is neg. if $\epsilon \leq \frac{1}{2^{80}}$
2. In thoery : ϵ is a function and $\epsilon : Z^{\geq 0} \rightarrow R^{\geq 0}$ and
   (a) ϵ is non-neg. if $\exists$ d : $\epsilon(\lambda) \geq \frac{1}{\lambda^d}$ infinitely often.
   (b) ϵ is neg. if $\forall$ d : $\epsilon(\lambda) \leq \frac{1}{\lambda^d}$

These definitions will start to make sense after a while in this report. Note that we still haven't answered the question of what it means for a stream cipher(using PRG) to be secure yet. For now, assume that a stream cipher is secure if it's PRG is secure and it's used "properly".

4.4 Attack on the OTP

We have said earlier that the OTP has perfect secrecy, and that was indeed true. However, the OTP is called "one-time" pad for a reason. The key cannot be used more than once! As it turns out the two-time pad is actually insecure and it lead to the US government discovering Russian secrets!

4.4.1 Attack 1 : the two-time pad

Here, the threat model is that the adversary has access to ciphertext, but not the key.

$$c_1 = m_1 \oplus PRG(k) \tag{13}$$
$$c_2 = m_2 \oplus PRG(k) \tag{14}$$
$$\Rightarrow c_1 \oplus c_2 = m_1 \oplus m_2 \tag{15}$$

As it turns out, there is enough redundancy in English and ASCII encoding that given $m_1 \oplus m_2$, we can recover both m_1 and m_2. Thus, Security is out of the window!

So the lesson we learn here is avoid related keys. If many keys have some part in common, we can wait for the algorithm to run through all frames, and once the frames start repeating, it is effectively a two-time pad. Moreover, related keys tend to make the adversary's life easy since, all keys are not pseudorandom anymore.

4.4.2 Attack 2 : the one-time pad

As it turns out, eventhough the OTP has perfect secrecy, the messages are malleable! That is, eventhough the adversary doesn't know the message, he can make sure that the message is mutated before it reaches the other party.

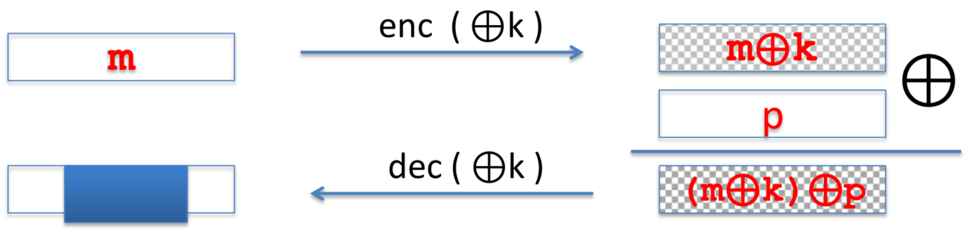


Figure 6: Malleability of OTP

Moreover, the change in ciphertext goes on undetected and the impact of the change is actually predictable. For instance, it is common to start emails with your name, so the adversary can change those bits only and leave the rest of the email untouched.

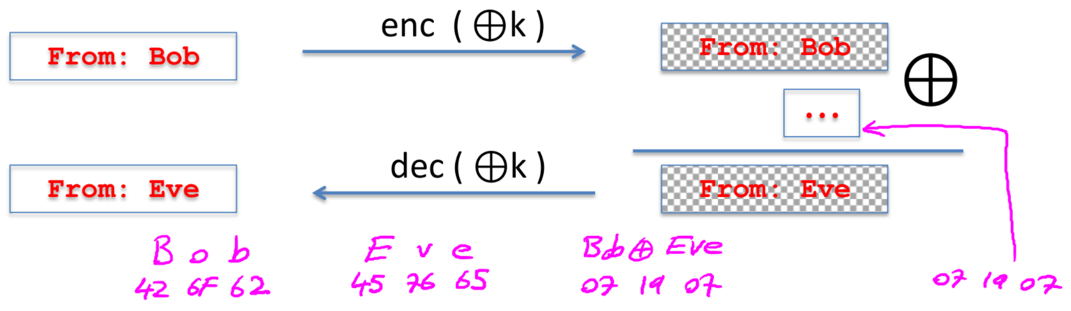


Figure 7: Example of OTP malleability

There are many examples of Stream Ciphers, however many of them fail under verious attacks. One excellent example of a Stream Cipher is Salsa20/12 (This has no known attacks!)

4.5 What is a Secure Cipher ?

Let G : K →{0,1}^{n} be a PRG. In this section our goal is to define what it means for

$$[k \overset{R}{\leftarrow} K : G(k)] \tag{16}$$

to be “indistinguishable” from

$$[r \overset{R}{\leftarrow} \{0,1\}^n : r] \tag{17}$$

But, before we begin, we need some definitions from discrete probability.

4.5.1 Statsitical Tests

Statistical test on $\{0,1\}^n$ an algorithm A such that, A(x) outputs 0 or 1.

For example,
A(x) = 1 iff $|\#0(x) - \#1(x)| \leq 10\sqrt{n}$
where #0(x) is the number of zeros in the binary string x and similarly #1(x) is the number of ones in the binary string.

4.5.2 Advantage

Let G : K $\rightarrow \{0,1\}^n$ be a PRG and A a statistical test on $\{0,1\}^n$ then,

$$Adv_{PRG}[A,G] := |Pr_{k \xleftarrow{R} K}[A(G(k)) = 1] - Pr_{r \xleftarrow{R} \{0,1\}^n}[A(r) = 1]| \in [0,1] \tag{18}$$

This definition basically quantifies how differently does a PRG behave from a random variable. If the advantage is close to 1 means that for that particular statistical test, PRG and true random variable act differently. If the advantage is close to 0, the PRG and true random variable act similarly for that test.

4.5.3 Secure PRGs

G : K $\rightarrow \{0,1\}^n$ is a secure PRG if $\forall$ efficient statistical tests A , $Adv_{PRG}[A,G]$ is "negligible".

Note that the existence of provably secure PRGs is unknown (the details are not included here but it is a P vs NP problem).

Let's start by proving an easy fact.
Lemma : A secure PRG is unpredictable, i.e. predictable PRG $\Rightarrow$ PRG is insecure.

Suppose A is an efficient algorithm such that,

$$Pr_{k \xleftarrow{R} K}[A(G(k))|_{1,\cdots,i} = G(k)|_{i+1}] = \frac{1}{2} + \epsilon \tag{19}$$

for non-negligible ϵ.

Now, we show that A breaks the PRG.
Define statistical test B as :

$$B(X) = \begin{cases} 1, & \text{if } A(X|_{1,\cdots,i}) = X_{i+1}. \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

$$r \xleftarrow{R} \{0,1\}^n \tag{21}$$

$$Pr[B(r)=1] = \frac{1}{2} \tag{22}$$

$$Pr[B(G(k))=1] \geq \frac{1}{2} + \epsilon \tag{23}$$

Thus, $Adv_{PRG}[B,G]$ is $\geq \epsilon$, which is non-negligible! Hence, the PRG is insecure.

Theroem (Yao '82) : An unpredictable PRG is secure

Theorem : If $\forall i \in \{0,\cdots,n-1\}$, G is unpredictable at position i then G is a secure PRG.
that is, if next-bit predictors cannot distinguish G from random, then no statistical test can! Now, let's further formalise the definition of a secure PRG using "computational indistinguishablility".

Definition : We say that P_1 and P_2 are computationally indistinguishable (denoted $P_1 \approx_P P_2$) if $\forall$ efficient statistical tests A,

$$|Pr_{x \leftarrow P_1}[A(x)=1] - Pr_{x \leftarrow P_2}[A(x)=1]| < \epsilon \tag{24}$$

where ϵ is negligible.

For example, A PRG is secure if $\{k \xleftarrow{R} K : G(k)\} \approx_P \{r \xleftarrow{R} \{0,1\}^n\}$. Note that, the secure PRG does not imply that a cipher is secure!

4.6 Semantic Security

Let us limit the attacker's abilities to only obtaining one ciphertext for now.

We can have the following security requirements. (attempts to quantify leaking of information)

1. Attacker cannot require secret key. However, this fails under the following encryption.
   E(k,m) = m. (Attacker can just take the ciphertext = plaintext here. No need of taking the key)

2. Attacker cannot recover all of plaintext. This fails due to the following encryption.
   E(k, $m_0||m_1$) $= m_0||k \oplus m_1$ (The attacker can use the "leaked m_0 to do a lot of damage)

So, we need a better way to define security. Shannon's perfect secrecy comes to mind. However, this is too strict of a criteria to be useful. Hence, we define a new criteria called Semantic Security!

For this, we first need a setup.

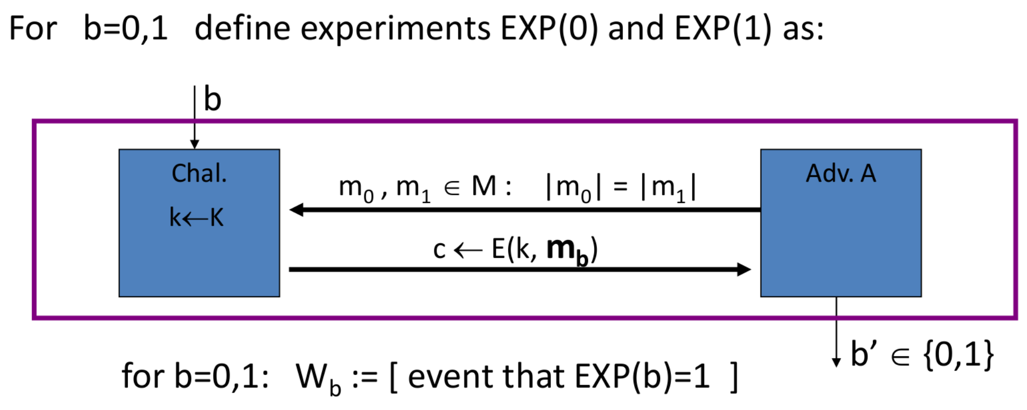

For b=0,1 define experiments EXP(0) and EXP(1) as:
b
Chal.
k←K
$m_0, m_1 \in M$: $|m_0| = |m_1|$
c ← E(k, m_b)
Adv. A
b' ∈ {0,1}
for b=0,1: W_b := [event that EXP(b)=1]

Figure 8: Modelling Adversary

$$Adv_{SS}[A, E] := |Pr[W_O] - Pr[W_1]| \in [0, 1] \tag{25}$$

where, W_i has been defined in Figure 8. Here, we are basically judging if the adversary can distinguish between m_0 and m_1 based on c_0 and c_1. Thus, the definition of semantic security is :

Definition : E is semantically secure if for all efficient A, $Adv_{SS}[A, E]$ is negligible.
For example, if the ciphertext "leaks" the least significant bit, and $LSB(m_0) = 0, LSB(m_1) = 1$ and a statistical test B gives the least significant bit of the ciphertext.

$$Adv_{SS}[B, E] = |[Pr(EXP(0) = 1)] - [Pr(EXP(1) = 1)]| = 1 \tag{26}$$

which is not negligible, hence, such a system is semantically insecure! It can be proven that an OTP is semantically secure. Even stream ciphers are semantically secure. (the proof is not covered here)

5 Block Ciphers

As we have studied in the previous section, stream ciphers are a good tool for secure transmission of messages. In this section, we will focus upon block ciphers and the interesting security solutions they give rise to, especially enabling many-time keys. Historically, one of the most famous block ciphers was DES, which was a result of a research project conducted by IBM and later adopted as a security standard by the US federal government. So let us dive into the construction of block ciphers!

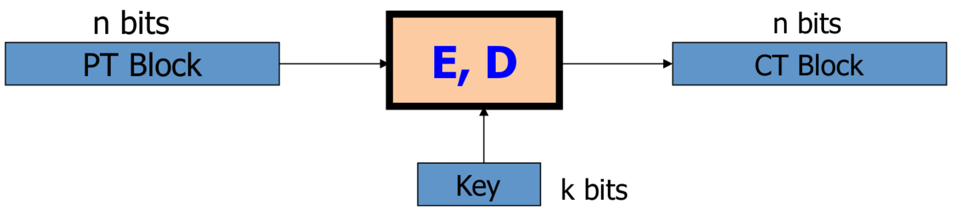


Figure 9: Block Cipher setup

5.1 Construction

The construction of a block cipher depends on mainly two parameters :

1. Key expansion mechanism
2. Round function

These factors define a block cipher which then takes a key and a plaintext block as input, outputing the ciphertext. R(k, m) represents the round function. The block cipher works using the process of iteration, the output of each round function acts as an input for the next round function, finally outputing the ciphertext.

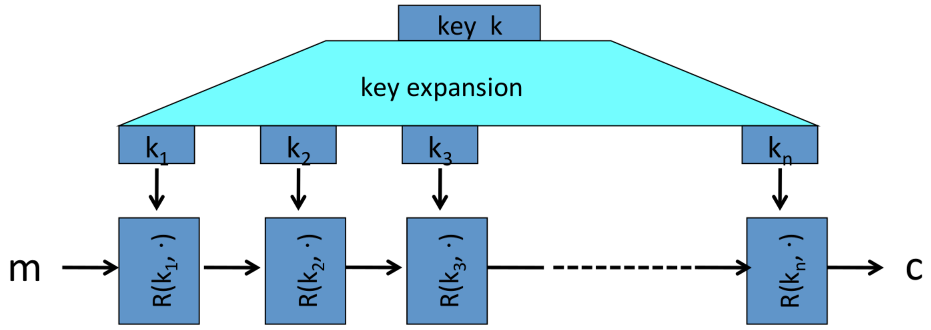
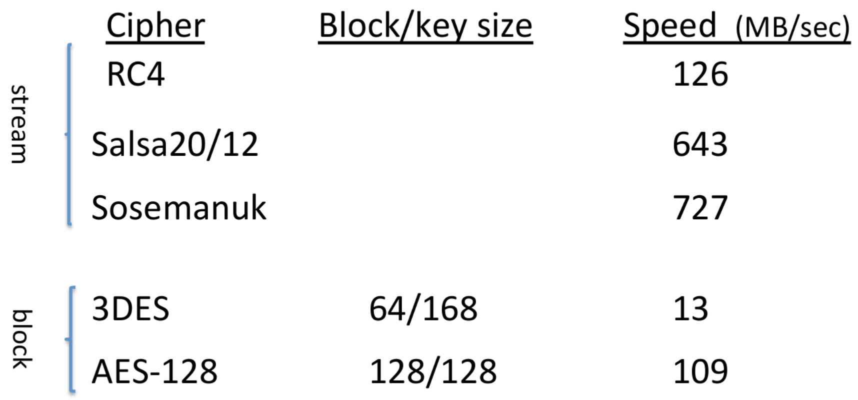


Figure 10: Block Cipher mechanism

Since, the block cipher adds so many more steps into the computation of a ciphertext block, we might expect it to be slower than stream ciphers, and this is indeed true. Following is the performance analysis on a 2.2 GHz AMD Opteron Processor (Linux)



	Cipher	Block/key size	Speed (MB/sec)
stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
block	3DES	64/168	13
	AES-128	128/128	109

Figure 11: Performance of ciphers

5.2 Abstraction : PRPs and PRFs

Since thinking about the mechanism of a block cipher everytime it is used can be difficult, we use an abstraction of the block cipher in the form of pseudo-random permutation and pseudo-random function.

A **pseudo-random function (PRF)** defined over (K, X, Y) :

$$F : K \times X \to Y \tag{27}$$

such that there exists an "efficient" algorithm to evaluate $F(k, x)$, where $k \in K$, $x \in X$.

A **pseudo-random permutation (PRP)** defined over (K, X) :

$$E : K \times X \to X \tag{28}$$

such that :

1. There exists "efficient" deterministic algorithm to determine $E(k, x)$, where $k \in K$, $x \in X$
2. The function $E(k, .)$ is one-one.
3. There exists an efficient inversion algorithm D(k, y)

We can also say that functionally, every PRP is a PRF with $Y = X$ and some additional properties.

This abstraction might sound confusing at first, it will become clearer when we arrive at the application of block ciphers. Basically this means that we can replace a block cipher with a pseudo-random permutation and forget about the internal mechanisms of a block cipher, that is, we can forget about key expansion, round function, etc.

5.3 Secure PRFs

Let $F : K \times X \to Y$ be a PRF.

$$\begin{cases} Funs[X, Y], & \text{set of all functions from X to Y.} \\ S_F = \{F(k, .) \mid k \in K\}, & \subseteq Funs[X, Y] \end{cases} \tag{29}$$

Similar to the logic covered for secure PRGs, our goal will be to make the distribution of PRFs indistinguishable from the distribution of all functions from X to Y.

Now, let us formalise the notion of a secure PRF further. For this purpose, we need (ref section 2.4) to specify the threat model. Here, the threat model

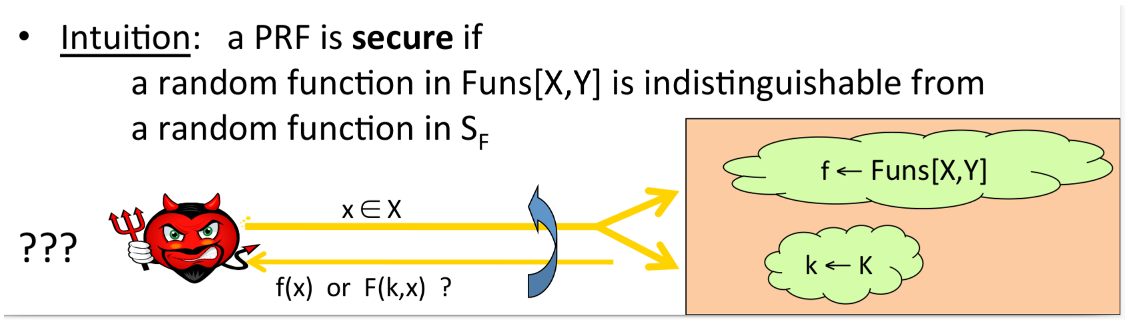

• Intuition: a PRF is **secure** if
a random function in Funs[X,Y] is indistinguishable from
a random function in S_F
???
$x \in X$
f(x) or F(k,x) ?
$f \leftarrow$ Funs[X,Y]
$k \leftarrow K$

Figure 12: Secure PRF

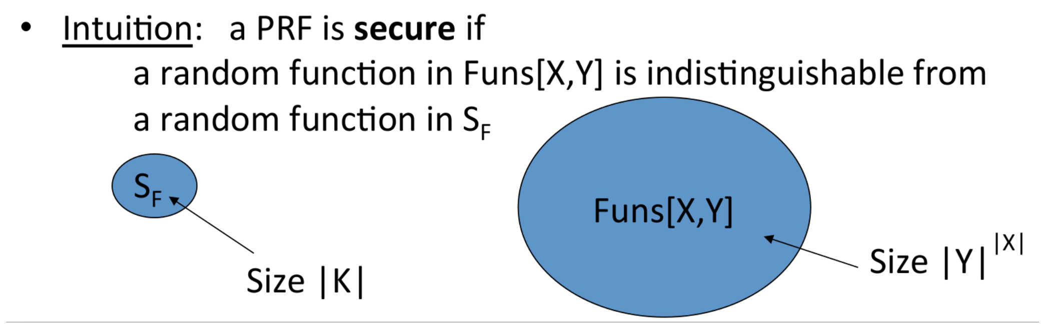

• Intuition: a PRF is **secure** if
a random function in Funs[X,Y] is indistinguishable from
a random function in S_F
S_F
Size |K|
Funs[X,Y]
Size $|Y|^{|X|}$

Figure 13: Secure PRF

is that the attacker can have multiple outputs for a given function and the attacker needs to determine whether the function he got is a truly random function or a PRF using some statistical test. If the advantage of the attacker is negligible for every statistical test, then our construction is secure. The schematic is given in the below diagram.

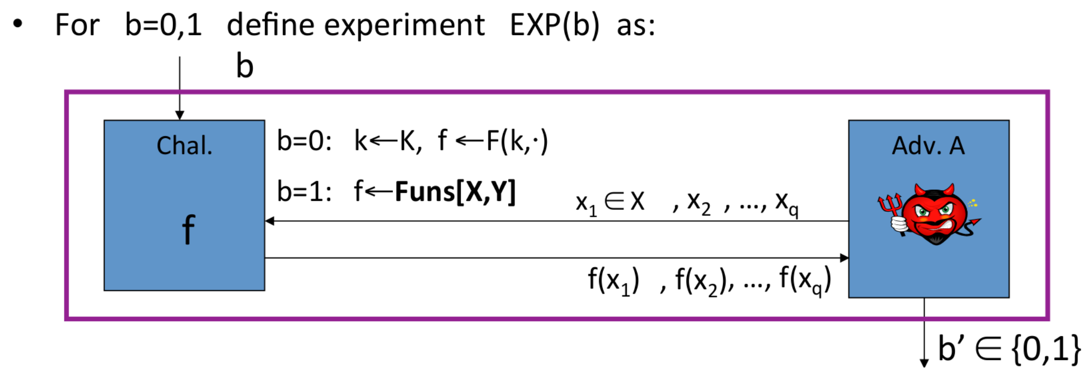


Figure 14: Modelling Adversary

Thus, E is a secure PRP if for all "efficient" A :

$$Adv_{PRF}[A, E] := | \, Pr[EXP(0) = 1] - Pr[EXP(1) = 1] \, | \tag{30}$$

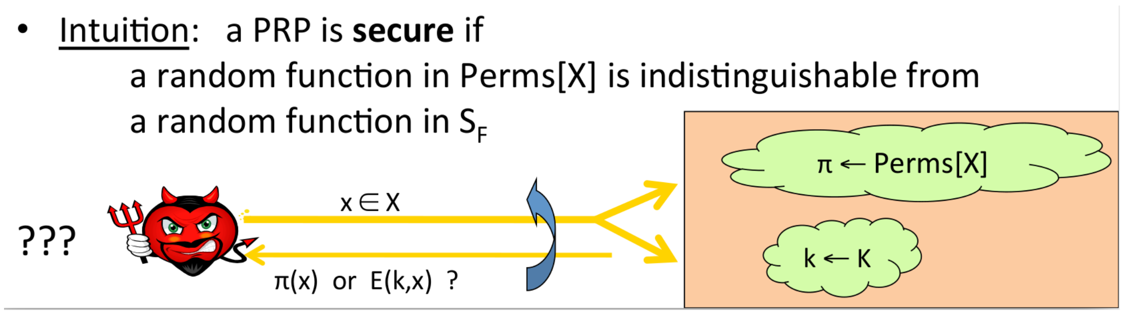
is negligible

5.4 Secure PRPs

Since, the PRPs are an abstraction of the block cipher, the definition of semantic security of a block cipher and a PRP are the same, as follows :
Let $E : K \times X \to X$ be a PRP,

$$\begin{cases} Perms[X], & \text{set of all one-one functions from X to X.} \\ S_F = \{E(k, .) \mid k \in K\}, & \subseteq Perms[X] \end{cases} \tag{31}$$

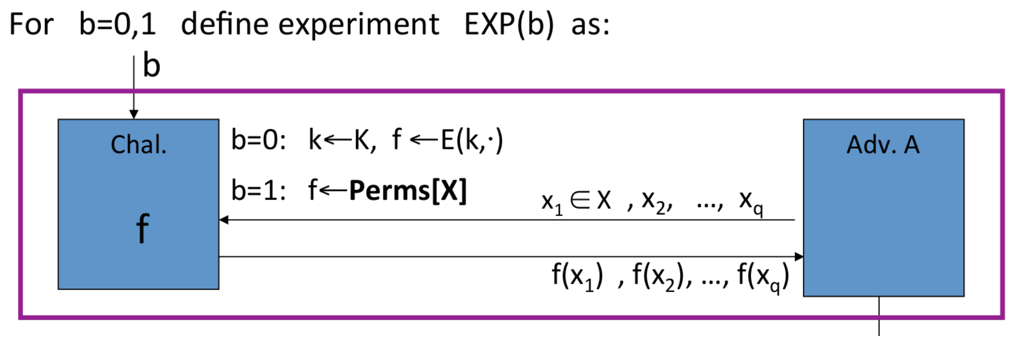
The reasoning of secure PRPs remains exactly the same as before, as demonstrated in the figure below.



• Intuition: a PRP is **secure** if
a random function in Perms[X] is indistinguishable from
a random function in S_F
???
$x \in X$
$\pi(x)$ or $E(k,x)$?
$\pi \leftarrow$ Perms[X]
$k \leftarrow K$

Figure 15: Secure PRP

Similar to PRFs, let us formalise the notion of security of PRPs. The threat model is that the attacker can have multiple outputs for a given one-one function and the attacker needs to determine whether the function he got is a truly random Permutation or a PRP using some statistical test. If the advantage of the attacker is negligible for every statistical test, then our construction is secure. The schematic is given in the below diagram.



For b=0,1 define experiment EXP(b) as:
b
Chal.
f
b=0: $k \leftarrow K$, $f \leftarrow E(k,\cdot)$
b=1: $f \leftarrow$ **Perms[X]**
$x_1 \in X$, x_2, …, x_q
$f(x_1)$, $f(x_2)$, …, $f(x_q)$
Adv. A

Figure 16: Modelling Adversary

Some examples of secure PRPs are :

1. 3DES
2. AES : $K \times X \rightarrow X$, where, $K = X = \{0,1\}^{128}$

An example concrete assumption about AES is that all 2^{80} time algorithms have $Adv_{PRP}[A,E] < \frac{1}{2^{40}}$

5.5 PRF switching lemma

Before introducing the PRF switching lemma, I would like to present a question.
Consider the 1-bit PRP, $E(k, x) = k \oplus x$

1. Is this a secure PRP ?
2. Is this a secure PRF ?

For the first question, we can observe that (0, 1) either maps to (1, 0) or (0, 1) depending upon the key k, and this is exactly the set of all possible one-one mappings. Hence, it is indeed a secure PRP. However, the second question does include mappings like (0, 1) to (0, 0) and (0, 1) to (1, 1) as well! Hence, this is NOT a secure PRF. If you want to check the definition for the second question you can use the statistical test :

$$B(f) = \begin{cases} 1, & \text{if } f(0) = f(1). \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

and the corresponding advantage turns out to be 1, which is non-negligible. Now that we know that every secure PRP is not a secure PRF, we can understand the significance of the PRF switching lemma.

Lemma : Every secure PRP is also a secure PRF, if $|X|$ is sufficiently large.

A more precise statement is :

Let E be a PRP over (K, X). Then for any q-query adversary A :

$$|Adv_{PRF}[A, E] - Adv_{PRP}[A, E]| < \frac{q^2}{2|X|} \tag{33}$$

where the RHS has to be "negligible".
Under these conditions, $Adv_{PRP}[A, E]$ being negligible ensures that $Adv_{PRF}[A, E]$ is also negligible and hence, guarantees that E is a secure PRF.

6 Operating Block Ciphers

Now that we have established a construction for a block cipher, and rigorous definitions regarding the security of PRPs and PRFs, we can start using block ciphers in different modes of operation.

6.1 One-time key

One-time keys are commonly seen in email encryption, where a new key is generated for every message. Here, our goal is to build a "secure" encryption from a secure PRP (like AES). The threat model for the same is as follows :

1. Adversary's power : Adversary can only see one ciphertext (one-time key)
2. Adversary's goal : Learn more information about plaintext from the ciphertext (breaking semantic security)

Let us formalise semantic security for one-time key or a one-time pad (OTP). Two experiments are performed as shown in the figure below :

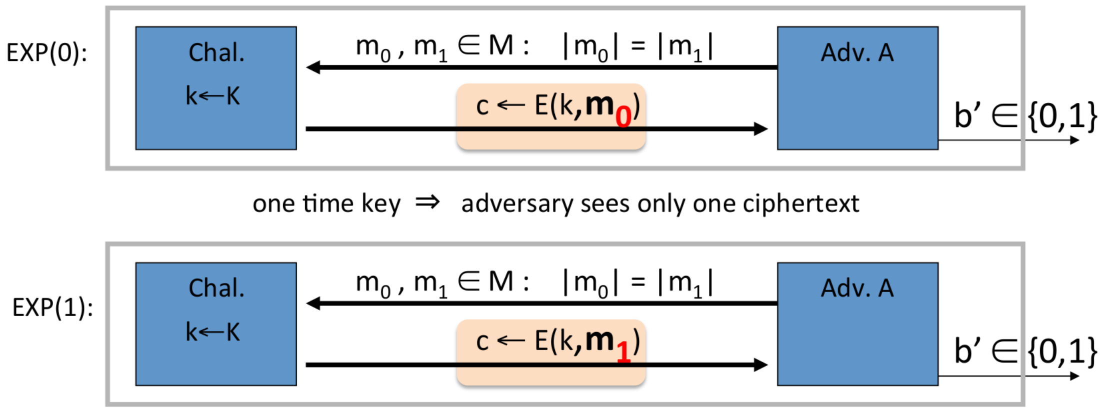


Figure 17: Semantic Security Game

A PRP $E : K \times X \to X$ is semantically secure if,

$$Adv_{SS}[A, OTP] = |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]| < \epsilon \tag{34}$$

where ϵ is "negligible".

By now, we are familiar with these kind of definitions. In this setup, only one ciphertext is available to the adversary.

6.1.1 Electronic Code-Book

A famous incorrect implementation of the one-time key is ECB (Electronic Code Book). In this setup, many plaintext blocks are encrypted using a block cipher with a one-time key and the corresponding ciphertext is obtained.

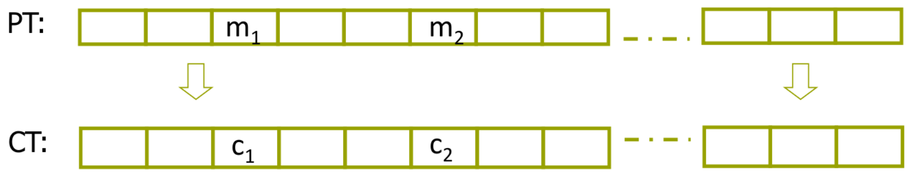


Figure 18: ECB

However, this is a badly broken cipher as it suffers from a critical flaw. The encryption of identical plaintext blocks gives exactly the same ciphertext block! Hence, using a CT-only attack, the adversary obtains information about the plaintext. This point is demonstrated in the following diagram, where all the "hair" in the image are encoded to the same vertical line, and the encrypted image clearly gives away information about the orignal image!

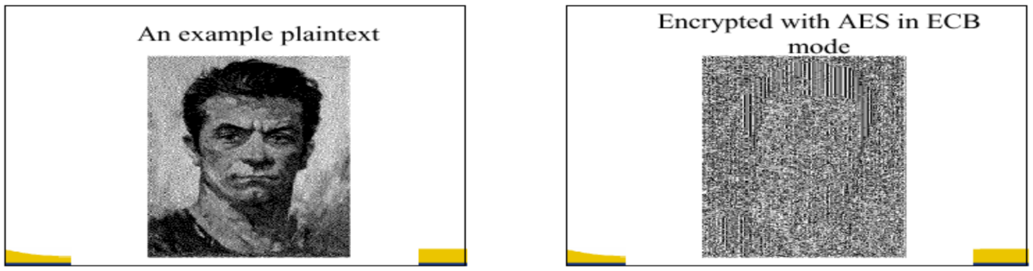


Figure 19: Example of ECB

6.1.2 Secure Construction I

The first secure construction is the deterministic counter mode from a PRF $F : K \times \{0,1\}^n \to \{0,1\}^n$, has a schematic as shown below :

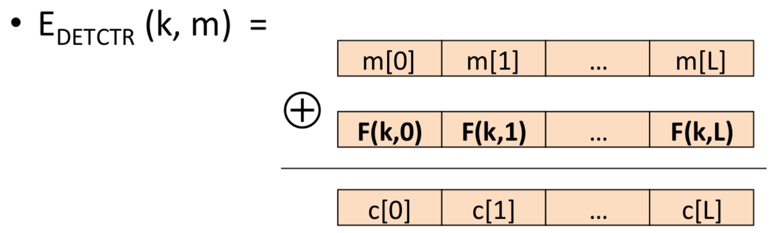


Figure 20: Deterministic Counter Mode

This is basically a stream cipher generated using a PRP (like 3DES, AES). In particular, we can state the following theorem about the security of a deterministic counter cipher.

Theorem :
For any $L > 0$, if F is a secure PRF over (K, X, X) then, E_{DETCTR} is a semantically secure cipher over (K, X^L, X^L). In particular, for any efficient adversary A attacking E_{DETCTR} there exists an efficient PRF adversary B such that :

$$Adv_{SS}[A, E_{DETCTR}] = 2 \times Adv_{PRF}[B, F] \tag{35}$$

Now, using this theorem, we know that $Adv_{PRF}[B, F]$ is negligible since F is a secure PRF, hence, we can say that E is a secure cipher! The sketch of a proof for the same is shown below.

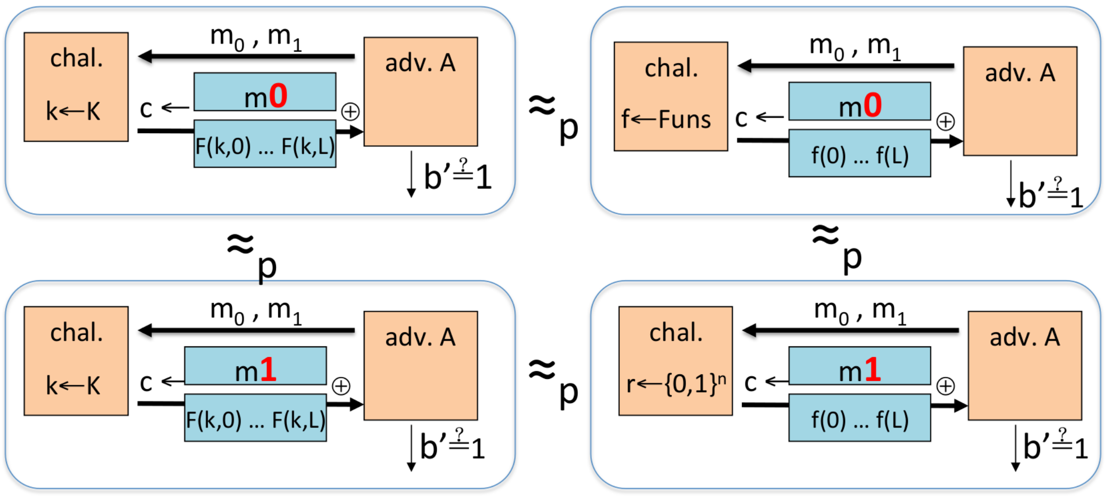


Figure 21: Proof of security

6.2 Many-time key

A Many-time key is something remarkable that can be achieved using a block cipher. In modern-day cryptography a many time key is used in the following :

1. File systems : Same AES key is used to encrypt many files
2. IP Security : Same AES key is used to encrypt many packets

First, we need a rigorous definition of semantic security for a many-time key. Later, we will take a look at some ideas to achieve semantic security for many-time keys and some constructions for many-time key systems.

6.2.1 Semantic Security for many-time keys

When dealing with security, the first thing we need to establish as always is a threat model and the adversary's goal. In a many-time key setup, the adversary can see many ciphertexts encrypted using the same key. Moreover, in real life adversary can actually have control over some of the plaintexts! For example, this can be achieved by the adversary sending an email to the person under attack and the person storing this email using disk encryption. Incorporating this, we get the following threat model :

1. Adversary's power : Chosen Plaintext Attack (CPA), that is, adversary can obtain the encryption of arbitrary messages of his choice.
2. Adversary's goal : Break semantic security

Now that we have established a threat model, we need a formal definition of semantic security. In some sense, for a many time key, the semantic security game is just a standard semantic security game just iterated over k queries, so that the attacker can adaptively issue queries one after the other. Now the chosen plain text attack is captured by the fact that if the attacker wants the encryption of a particular message m. What he could do is, use query j for some j, and in this query j he'll set both the first message and the second message to be the exactly same message m. In other words, both the first message and the second message are the same, and both are set to the message m. In this case, he will receive the encryption of this message m that he was interested in.

Definition (CPA Security) : Let $E = (E, D)$ be a cipher defined over (K, M, C). For b = 0, 1 define EXP(b) as the following process :

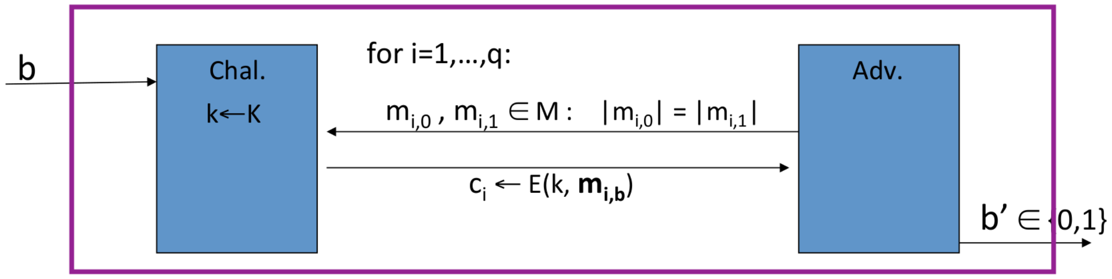

b
Chal.
k←K
for i=1,…,q:
$m_{i,0}$, $m_{i,1}$ ∈ M : $|m_{i,0}| = |m_{i,1}|$
$c_i \leftarrow E(k, m_{i,b})$
Adv.
b' ∈ {0,1}

Figure 22: CPA Security

where, adversary queries $m_{j,1} = mj, 0 = m$ if he wants $c = E(k, m)$. Now, E is semantically secure under CPA if for all "efficient" A :

$$Adv_{CPA}[A, E] = |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]| \tag{36}$$

is negligible

6.2.2 Ciphers insecure under CPA

It happens that ciphers (deterministic) are always insecure! We can prove this as follows,

Suppose $E(k, m)$ outputs the same ciphertext for the message m, then the adversary can issue the queries as shown below :

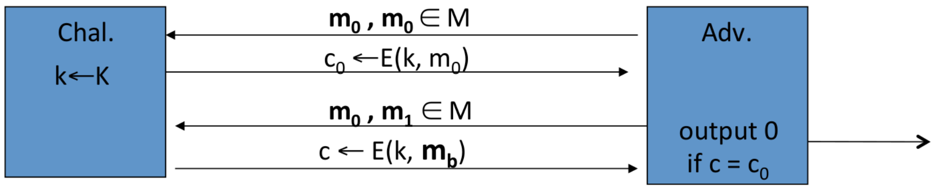


Figure 23: Breaking Ciphers

thus, the adversary can distinguish between event 1 and event 0 with certainty, implying a non-negligible advantage breaking semantic security.

The fundamental issue in ciphers is that the outputs are deterministic, that is, the same plaintext maps to the same ciphertext, hence cannot be used for many-time keys. This problem is solved using the tools which we will establish in the next few sections.

6.2.3 Solution 1 : randomised encryption

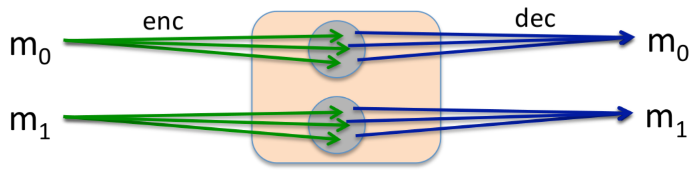


Figure 24: Randomised Encryption

In a randomised encryption scheme, $E(k, m)$ is a randomised algorithm :

1. Encrypting same message twice gives different ciphertexts.
2. Ciphertext must be longer than plaintext. Roughly speaking, CT-size = PT-size + #random bits

Say Random bits = additional 128 bits. For messages of size GB this doesn't matter, but if plaintext is itself small this is a lot of unnecessary junk, causes increased cost.

6.2.4 Solution 2 : Nonce-based Encryption

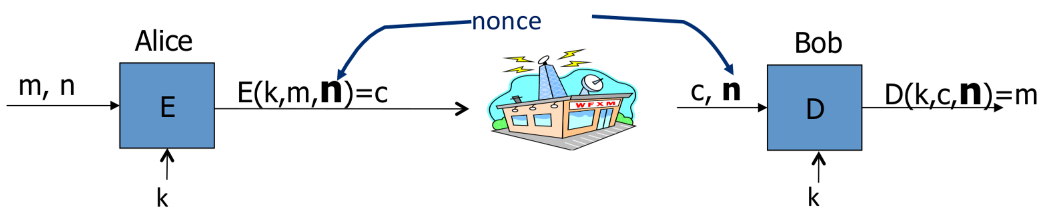


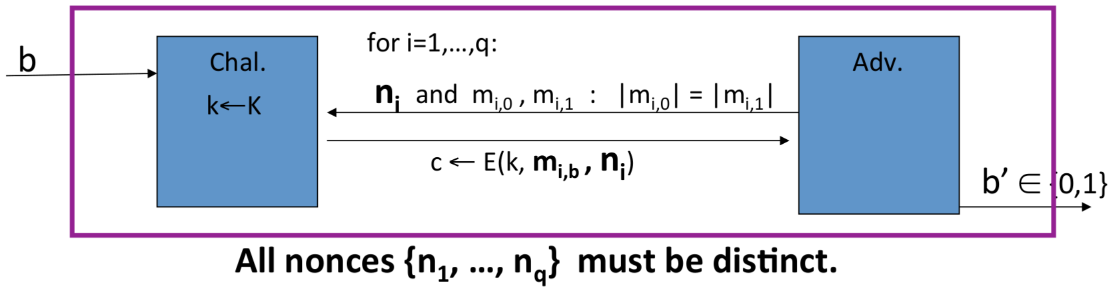
Figure 25: Nonce-based Encryption

A nonce "n" is a value that changes from message to message, and the pair (k, n) is never used more than once. The nonce can be used in two methods.

1. nonce is a counter
   (a) Used when encryptor keeps state from msg to msg
   (b) if decryptor has same state, nonce need not be contained in the ciphertext
2. Encryptor chooses a random nonce $n \leftarrow N$, where N is the nonce-space.

Semantic security for a nonce-based encryption is exactly the same as CPA, but the nonce is chosen by the adversary and all the nonces must be distinct (since, nonce can never repeat while encryption).

6.2.5



b
Chal.
k←K
for i=1,…,q:
$\mathbf{n_i}$ and $m_{i,0}$, $m_{i,1}$: $|m_{i,0}| = |m_{i,1}|$
c ← E(k, $\mathbf{m_{i,b}}$, $\mathbf{n_i}$)
Adv.
b' ∈ {0,1}
All nonces {n_1, …, n_q} must be distinct.

Figure 26: Nonce-based Semantic Security

Nonce-based E is semantically secure under CPA if ∀ "efficient" adversary A :

$$Adv_{nCPA}[A, E] = |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]| \tag{37}$$

is negligible

6.3 Modes of operation : Many-time keys

Now that we have an idea of various ideas to ensure semantic security, we will have a look at some cryptographic constructions.

6.3.1 Cipher-block Chaining(CBC)

The first construction we will have a look at is called **CBC with random IV**, where IV is the initialisation vector.

Let (E, D) be a PRP. $E_{CBC}(k, m)$ chooses a random IV $\in X$, where

$$E : K \times \{0,1\}^n \to \{0,1\}^n \tag{38}$$

$$X = \{0,1\}^n \tag{39}$$

The schematic for the same is given below :

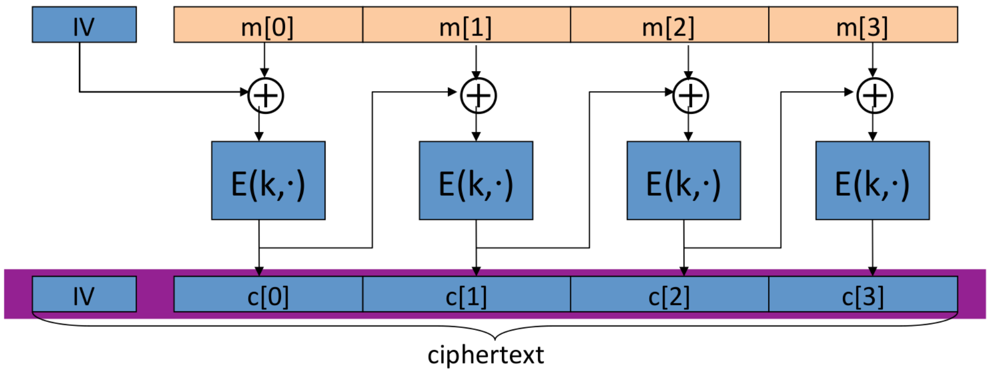


Figure 27: Encryption Circuit

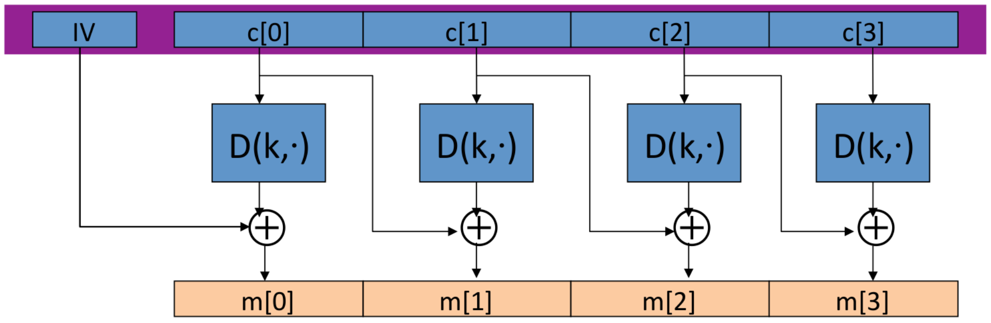


Figure 28: Decryption Circuit

6.3.2 CTR Mode

The first construction under this is called **random-ctr mode**.
Let $F : K \times \{0,1\}^n \to \{0,1\}^n$ be a secure PRF. $E(k,m)$ uses a random IV as follows,

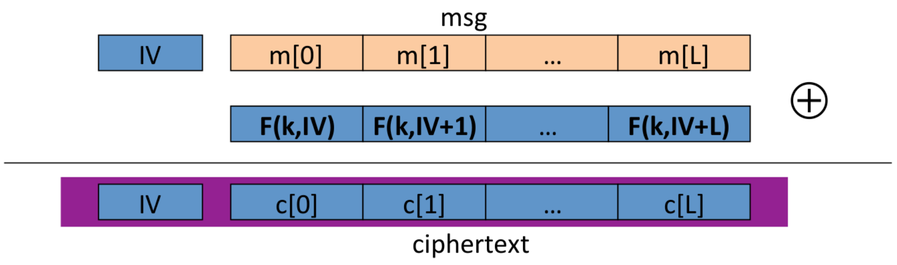
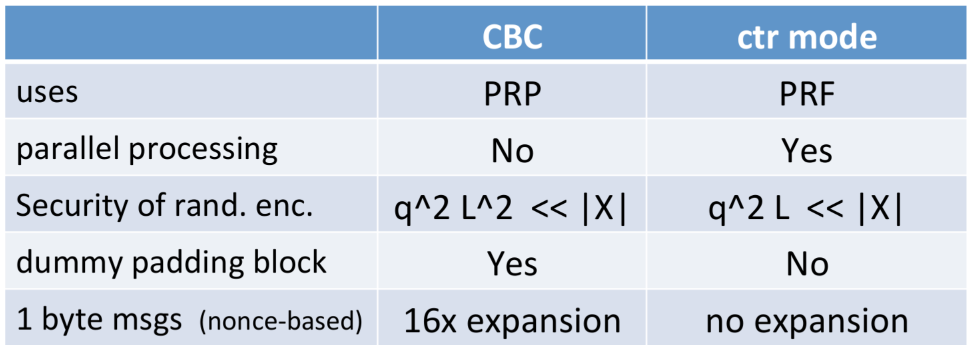


Figure 29: Random-CTR

Note that, this mode is parallelizable unlike CBC which is inherently sequential in nature!

6.3.3 CTR vs CBC

In conclusion, this table summarizes the two methods of constructing many-time keys.



	CBC	ctr mode
uses	PRP	PRF
parallel processing	No	Yes
Security of rand. enc.	q^2 L^2 << |X|	q^2 L << |X|
dummy padding block	Yes	No
1 byte msgs (nonce-based)	16x expansion	no expansion

Figure 30: Comparison

7 Concluding remarks

In this report, I covered basic aspects of security, how cryptography has evolved through the years. If anyone is interested in learning more about cryptography, Prof. Dan Boneh has an excellent course named "Cryptography-I" (which happens to be the course I am studying from as well). I have tried to make stuff more intuitive in this report and have tried to include historical references since that gives a certain "humane" nature to the problems of cryptography. Moreover, we can see the application of number theory here in the form of modular arithmetic, XOR, random distributions and probability. In conclusion, thank you for reading this report and keep on exploring !

8 Bibliography

References

[1] Yao's paper : https://research.cs.wisc.edu/areas/sec/yao1982-ocr.pdf

[2] Dan Boneh's Course : https://www.coursera.org/learn/crypto