1 Confidentiality

Confidentiality can be achieved by many cryptographic method and techniques, for example, Symmetric Key Encryption, Asymmetric Key Encryption and Digital Signatures. In this section, we will focus our attention on Symmetric Key Encryption (SE).

1.1 Symmetric Key Encryption (SE)

Symmetric Encryption consists of the following algorithms (Key Generation, Encryption and Decryption).

- Key Generation ($KeyGen()$)
- Encryption ($Enc(m, k)$)
- Decryption ($Dec(c, k)$)

1. Key Generation is simply a randomized algorithm that generates a key for the encryption process. The randomized key is said to be unique and random. The Dollar Symbol ($) below represents the randomized generation of the unique key.

   \[ k \leftarrow \text{KeyGen()} \text{ where } k \text{ is the unique generated key} \]

2. Encryption is a randomized algorithm that takes the input of $m$ and $k$ (plaintext message and key, respectively) and produce the output $c$ as the ciphertext. Due to the fact that $k$ is random and unique, the output of $c$ is also unique everytime.

   \[ c \leftarrow \text{Enc}(m, k) \text{ where } c \text{ is the ciphertext generated by the combination of the plaintext } m \text{ and key } k \]

3. Decryption is a deterministic algorithm which takes the input of ciphertext $c$ and key $k$ and reproduce the original plaintext message $m$ as the output. Given that
this is Symmetric Key Encryption, the same exact key $k$ must be used for the decryption process.

$$m \leftarrow \text{Dec}(c, k)$$ where $m$ is the plaintext message

### 1.2 Properties of Symmetric Key Encryption

In the definitions and theorems below, we refer to probability distributions over $\mathbb{K}$, $\mathbb{M}$, and $\mathbb{C}$. The distribution over $\mathbb{K}$ is simply the one that is defined by running $\text{KeyGen}$ and taking the output. For $k \in \mathbb{K}$, we let $Pr(K = k)$ denote the probability that the key output by $\text{KeyGen}$ is equal to $k$. (Formally, $K$ is a random variable denoting the value of the key.) Similarly, for $m \in \mathbb{M}$ we let $Pr(M = m)$ denote the probability that the message is equal to $m$. For $c \in \mathbb{C}$ we let $Pr(C = c)$ denote the probability that the ciphertext is equal to $c$.

#### 1.2.1 Correctness Property

If the Encryption of $(m, k)$ yield $c$, then Decryption of $(c, k)$ should yield $m$.

If $c \xleftarrow{\$} \text{Enc}(m, k)$ then $m \leftarrow \text{Dec}(c, k) \quad \forall m \in \mathbb{M}, k \in \mathbb{K}$

#### 1.2.2 Security Property

An Encryption Scheme is said to be perfectly secure if it cannot be broken by an adversary who possesses unbounded computational power.

**Definition 1 (Perfectly Secure Encryption)** A Symmetric Encryption (SE) is called perfectly secure if:

$$Pr(M = m) = Pr(M = m|C = c) \quad \forall m \in \mathbb{M}, k \in \mathbb{K}, c = \text{Enc}(m, k)$$

where the probability of eavesdropping on the encrypted message cannot make out the message is said to be just as good as having "no knowledge" of the original message. With that, it is said to be perfectly secure.

**Definition 2 (Perfectly Secure Encryption)** A Symmetric Encryption (SE) is called perfectly secure if:

$$Pr(C = c) = Pr(C = c|M = m)$$
Based on Bayes’ Theorem \( \Pr(A|B) = \left( \frac{\Pr(A \cap B)}{\Pr(B)} \right) \)

**Definition 1 ⇔ Definition 2**

**Proof.** \( \Pr(M = m) = \Pr(M = m|C = c) \)

\( \iff \Pr(M = m) = \left( \frac{\Pr(M = m \cap C = c)}{\Pr(C = c)} \right) \)

\( \iff \Pr(C = c) = \left( \frac{\Pr(C = c \cap M = m)}{\Pr(M = m)} \right) = \Pr(C = c|M = m) \) where,

\( \Pr(C = c) \) - probability of guessing blindly what can be ciphertext \( c \) and

\( \Pr(C = c|M = m) \) - probability of guessing what can be the value of ciphertext \( c \) for the known message \( m \).

With that, the two definitions are said to be equivalent.

**Definition 3 (Perfectly Secure Encryption)** A symmetric encryption is said to be perfectly secure if:

\[ \Pr(C = c|M = m_0) = \Pr(C = c|M = m_1) \quad \forall m_0, m_1 \in \mathbb{M}, c \in \mathbb{C}, c = \text{Enc}(m, k) \]

where,

\( \Pr(C = c|M = m_i) \) - probability of guessing what can be the ciphertext \( c \) for the message \( m_i \), where \( i = 0, 1 \).

This indicates that if a sender encrypts two messages \( m_0 \) and \( m_1 \), then it is difficult to guess ciphertext \( c \) belongs to which message.

Thus, three above definitions can be shown to be equivalent. In other words, 

**Definition 1 ⇔ Definition 2 ⇔ Definition 3**

### 1.3 Construction

**One Time Pad (OTP): A Perfectly Secure Encryption Scheme.** This encryption scheme provides perfect secrecy. The message space (\( \mathbb{M} \)), key space (\( \mathbb{K} \)) and ciphertext space (\( \mathbb{C} \)) are all of the same size. We assume that the message space consists of \( l \)-bit long strings. OTP consists of:

- **Key Generation Algorithm:** A randomized algorithm that outputs a random key.
  
  \(- k \xleftarrow{\$} \{0, 1\}^l \)

- **Encryption:** The ciphertext is obtained by XOR operation of message \( m \) with key \( k \) i.e.
Input: $m \in \{0, 1\}^l$

Output: $c = m \oplus k$

- Decryption: The message $m$ is retrieved by XOR operation of ciphertext $c$ with key $k$ i.e.
  
  $m = c \oplus k$

**Theorem 1** One Time Pad (OTP) is perfectly secure

**Proof.** Let $m_0, m_1 \in \mathbb{M}$, $c \in \mathbb{C}$ and $k \in \mathbb{K}$. Also, $c = m \oplus k$ where $\mathbb{K} = \{0, 1\}^l$ and $l$ (l-bit) is the length of the key.

$\forall m_0 \in \mathbb{M}$

$Pr(C = c | M = m_0)$

$= Pr(M \oplus K = c | M = m_0)$

$= Pr(m_0 \oplus K = c)$

$= Pr(K = m_0 \oplus c)$

$= \frac{1}{2^l}$

$\forall m_1 \in \mathbb{M}$

$Pr(C = c | M = m_1)$

$= Pr(M \oplus K = c | M = m_1)$

$= Pr(m_1 \oplus K = c)$

$= Pr(K = m_1 \oplus c)$

$= \frac{1}{2^l}$

This means that $Pr(C = c | M = m_0) = Pr(C = c | M = m_1) = \frac{1}{2^l}$. Hence we can conclude that OTP is perfectly secure.

**Example to illustrate perfect secrecy of OTP:** Let $l = 2$. Therefore, $\mathbb{M} = \mathbb{K} = \mathbb{C} = \{0, 1\}^2 = \{00, 01, 10, 11\}$. Given that the ciphertext $c = 10$, let $m_0 = 00$ and $m_1 = 10$. We know that, $c = m \oplus k$.

$m_0 \oplus k = c$

00 $\oplus$ 00 = 00
00 $\oplus$ 01 = 01
00 $\oplus$ 10 = 10
00 ⊕ 11 = 11

Pr(C = c|M = m_0) = \frac{1}{4}

m_1 ⊕ k = c
10 ⊕ 00 = 10
10 ⊕ 01 = 11
10 ⊕ 10 = 00
10 ⊕ 11 = 01

Pr(C = c|M = m_1) = \frac{1}{4}

Pr(C = c|M = m_0) = Pr(C = c|M = m_1) = \frac{1}{4}

It can be seen from the above example that a single ciphertext '10' is generated with a probability of \(\frac{1}{4}\) for both \(m_0\) and \(m_1\). Hence we can conclude that OTP is perfectly secure for this example.

The scheme is very efficient because ⊕ is the only operation and its is proved to be perfectly secure.

It has a drawback because no of bits in key space is equal to no of bits in message space. Hence it does not work in practise.

**Theorem 2** For any perfectly secure encryption scheme, it must be the case that \(|K| \geq |M|\), i.e., the key space is at least as big as the message space.

**Proof.** Let us assume that a perfectly secure encryption scheme yielded a ciphertext \(c\) such that \(c \leftarrow Enc(m, k)\). Given \(c\), an adversary trying to break the scheme will compute all possible candidate plaintext values \(m_i\) using all possible keys \(k_i\) such that \(m_i = Dec(c, k_i)\)

If we assume that the key space is smaller than the message space, i.e., if \(|K| < |M|\), then there will not be complete mappings from key space \(K\) to message space \(M\) as some message space will be left out. Therefore,

\[ Pr(Enc(m_i, k_i) = c) = 0 \quad \text{for some } m_i \in M \text{ and } \forall k_i \in K \]

However, for the remaining messages \(m_i\) such that \(m_i = Dec(c, k_i)\), the probabilities are non-zero. Therefore, all messages are not equally likely to be the possible plaintext corresponding to a given ciphertext. This violates the property of a perfectly secure encryption (from Definition 3) and thus leads to a contradiction.

**Example to illustrate above theorem:** Let \(M = C = \{0, 1\}^2 = \{00, 01, 10, 11\}\) and \(K = \{00, 01\}\). Here we selected a scenario in which \(|K| < |M|\). Given that the ciphertext
Figure 1: When Key Space is less than Message Space

\[ c = 10, \text{ let } m_0 = 00 \text{ and } m_1 = 10. \text{ We know that, } m \oplus k = c. \]

\[
\begin{align*}
m_0 \oplus k &= c \\
00 \oplus 00 &= 00 \\
00 \oplus 01 &= 01 \\
Pr(C = c|M = m_0) &= 0
\end{align*}
\]

\[
\begin{align*}
m_1 \oplus k &= c \\
10 \oplus 00 &= 10 \\
10 \oplus 01 &= 11 \\
Pr(C = c|M = m_1) &= \frac{1}{2}
\end{align*}
\]

\[ Pr(C = c|M = m_0) \neq Pr(C = c|M = m_1) \]

It can be seen from the above example that \( c \) cannot take the value 10 for \( m_0 \) for the given key space \( K = \{00, 01\} \). So, \( Pr(C = c|M = m_0) = 0 \). On the other hand, for \( m_1 = 10 \), \( Pr(C = c|M = m_1) = \frac{1}{2} \) for the given key space \( K = \{00, 01\} \). This clearly violates Definition 3 (under Perfectly Secure Encryption), we can conclude that SE is not perfectly secure. Therefore, this contradicts our assumption. Hence, for perfect security, \(|K| \geq |M|\).

Since a perfectly secure encryption requires key space to be at least as big as the message space, it would not be possible to use a perfectly secure encryption scheme in practice. This motivates us to consider a weaker yet practical adversary – the one whose computation is bounded.

# 2 Computational Security

## 2.1 Computationally bounded adversary

**Definition 4 (Computationally Secure Primitive)** A cryptographic primitive will be called computationally secure if it is secure against an adversary which is computa-
tionally bounded.

2.2 Asymptotic Lower Bound: ”O” Notation

Definition 5 (Asymptotic Upper Bounded Function) A function \( f(n) \) which takes input parameter \( n \) is said to be asymptotic upper bounded by \( g(n) \) (i.e. \( f(n) = O(g(n)) \)) if:

\[ \exists c \text{ and } n_0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0 \]

As \( n \) becomes very large, \( f(n) \) is upperbounded by \( cg(n) \).

![Figure 2: Asymptotic Lower Bound](image)

2.3 Polynomial Time Algorithm

Definition 6 (Polynomial Time Algorithm) An algorithm which runs on an input parameter \( n \), is said to be a Polynomial Time Algorithm, if \( T(n) = O(n^k) \) for a constant \( k > 0 \), where \( T(n) \) is the time for computation and \( \exists c \text{ and } n_0 \text{ such that } T(n) \leq cn^k, \forall n \geq n_0 \)

Since the adversary has limited computation power, hence it must be bounded by polynomial time.

For exponential, \( T(n) = O(2^n) \).

2.4 Negligible Functions

Definition 7 (Negligible Function) A function \( P(n) \) running on input parameter \( n \) is said to be negligible if \( \forall c \exists n_0 \text{ such that} \)

\[ P(n) \leq \frac{1}{n^c}, \forall n \geq n_0 \]
As $n$ becomes very large, $P(n)$ is bounded by very small value. $n \to \infty$, $f(n) \to 0$.

System is secure, if the adversary runs in polynomial time but the probability of success is negligible.

2.5 Computational Security

**Definition 8 (Computationally Secure Cryptographic Primitive)** A cryptographic primitive will be called computationally secure if it is secure against an adversary which is run in a polynomial amount of time. In other words, for all adversary who run in polynomial amount of time success probability must be negligible.

**Example:** The DES encryption algorithm is represented as $c = DES(k, m)$. A brute force algorithm works as follows:

```plaintext
for $k_i = 0, 1, 2, \ldots, 2^n$
{
    if ($c == Enc(m, k_i)$)
        return $k_i$
}
```

Hence $T(n) = O(2^n)$, $p(n) = 1$ which is not a good adversary.
But if algorithm is modified as follows

\[
\text{for } k_i = 0, 1, 2, \ldots, n
\{
\text{if } (c = \text{Enc}(m, k_i))
\quad \text{return } k_i
\}
\]

then \( T(n) = O(2^n) \), \( p(n) = \frac{n}{2^n} \) since

\[
Pr(sum) = Pr(E_1 \cup E_2 \cup \ldots \cup E_n)
\]
\[
Pr(sum) \leq Pr(E_1) + Pr(E_2) + \ldots + Pr(E_n)
\]
\[
Pr(sum) = \frac{1}{2^n} + \frac{1}{2^n} + \frac{1}{2^n} \ldots = \frac{n}{2^n}
\]