1 Confidentiality

Refer to Lecture 1.

1.1 Authentication

Authentication is a way of proving that you are indeed who you are claiming to be. (e.g. Prove your identity). Bob receives a message from Alice and wants to guarantee that it was in fact generated by Alice.

1.2 Ways to Provide Authentication

1.2.1 Encryption for Authentication (Symmetric-Key Based)

Assume that Alice and Bob share a key $K$ with each other

- In order to authenticate a message $M$ to Bob, Alice will send the message and the encryption of the message ($M, C = Enc_K(M)$).

- Bob will decrypt $C$ (the ciphertext) and compare if it is the same as the message ($M == Dec_K(C)$).

Since, Bob knows that only Alice posseses the key $K$, therefore only Alice could have produced a valid encryption. In other words, an adversary not knowing the key, would not be able to impersonate Alice. This is the simplest security notion for an authentication scheme. We will be studying this and other notions in detail.

There is another faster symmetric-key based mechanism to provide authentication. It is called Message Authentication Code (MAC) and we will be studying it later in the course.

1.2.2 Encryption for Authentication (Asymmetric-Key Based)

- Alice will send the message to Bob (e.g. “I am Alice”).
• Bob will challenge by encrypting a random number $R$ using Alice’s public key and sending it to Alice. $(Enc_{PK_A}(R))$

• Alice will use her private key to decrypt the message and respond with $R$

Since, only Alice could have decrypted the challenge, Bob has the guarantee about Alice’s authenticity.

### 1.2.3 Digital Signature (Asymmetric-Key Based)

- **Key Generation**: Both Alice and Bob generate a secret key and a public key pair. That is, Alice generates $(SK_A, PK_A)$ and Bob generates and $(SK_A, PK_A)$

- **Signing**: Alice signs the message $M$ using her secret key and sends the signature $(S)$ and the message $(M)$ to Bob.
  1. Input: $(M, SK_A)$
  2. $S ← Sign_{SK_A}(M)$
  3. Output: $(M, S)$

- **Verification**: Bob receives the signature $(S)$ and the message $(M)$ and uses Alice’s public key to verify that the message is indeed signed by Alice and thus authentic.
  1. Input: $(M, S, PK_A)$
  2. Accept/Reject $← Verify_{PK_A}(M, S)$
  3. Output: $(Accept/Reject)$

### 2 Integrity

Integrity is the correctness of the data. We need a method to ensure that the message was not altered while in transit, that is, the message Bob receives is the same one that Alice intended to send. Generally, authentication and integrity go hand-in-hand. The mechanisms used to provide authentication can be used to provide integrity.

### 3 Non-repudiation

Alice should not be able to deny (repudiate) sending the message. We need to guarantee that Alice should not be able to deny what she previously admitted/commited to.
3.1 Techniques for Providing Non-repudiation

1. Using digital signatures (exactly as described in case of authentication). Nobody but Alice knows her secret key and thus when Bob receives a message and signature pair that verifies using Alice’s public key, this guarantees that Alice and only Alice has signed the message.

2. Symmetric-key encryption and Message Authentication Code can not be used for non-repudiation because both parties have the secret key and either of them produce valid MACs or encryption on behalf of the other.

4 Availability

Availability is the ability to have a communication channel between Bob and Alice. For example, an attacker could jam the communication channel or launch a denial-of-service attack on the communication channel. Cryptography can not solve the problem of availability, but it could help improve availability in certain applications.
Symmetric Key Encryption (SE)

Symmetric Key Encryption consists of the following triplet of algorithms:

- Key Generation
- Encryption
- Decryption

All the above run on an input of a security parameter ‘l’

1. Key Generation: A randomized algorithm (that uses “coin flipping”), which produces a random key of length l. A different key is generated every time the algorithm is run. The input, in this case, for the algorithm is null. The randomized output is represented with a dollar symbol ($). Let’s denote the keyspace as $\mathcal{K}$:

   (a) Input: Null
   (b) $k \leftarrow KeyGen()$

2. Encryption: A randomized algorithm that takes a plaintext $m$ and the key $k$ as an input, and produces the cipher text $c$ as the output.

   (a) Input: $m \in \mathcal{M}, k \in \mathcal{K}$ (where $\mathcal{M}$ is the message space)
   (b) $c \leftarrow Enc(m, k)$ (otherwise $\phi$ if $m \notin \mathcal{M}$)  
      
      $m \in C$, where $\mathcal{C}$ is the ciphertext space

3. Decryption: This is a deterministic algorithm.

   (a) Input: $c \in \mathcal{C}, k \in \mathcal{K}$
   (b) Output: $m \leftarrow Dec(c, k)$

4.0.1 Correctness Property

It ensures that if message is encrypted using a key, then the same message is returned as the output when the cipher text is decrypted i.e. If $c \leftarrow Enc(m, k)$ then $m \leftarrow Dec(c, k) \quad \forall m \in \mathcal{M}, k \in \mathcal{K}$
4.0.2 Security Property

We study the properties an SE should possess in order to be securely used for achieving confidentiality. In other words, we study various notions of security of an SE.

The first property that we study is called perfect or unconditional or information-theoretic security. In achieving perfect security, we consider an adversary who has an unbounded computational power (i.e. it possesses infinite amount of computing power to break an SE).

**Notation:** $Pr(X = x)$ denotes the probability that a random variable $X$ equals a value $x$. For example, $Pr(M = m)$ denotes the probability that the encrypted messages equals $m$; $Pr(K = k)$ denotes the probability that the key picked by the Key Generation algorithm equals $k$; $Pr(C = c)$ denotes the probability that the ciphertext equals $c$;

**Definition 1 (Perfect Security)** A symmetric encryption is said to be perfectly secure if \( \forall m \in M, c \in C \), and for all probability distributions over $M$,

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

Intuitively, this means that in a perfectly secure encryption, the knowledge of cipher text is as good as no knowledge of it.

**Definition 2 (Perfect Security)** A symmetric encryption is said to be perfectly secure if \( \forall m \in M, c \in C \), and for all probability distributions over $M$,

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

The two definitions can be shown to be equivalent. In other words, Definition1 $\Leftrightarrow$ Definition2. (Hint: Use Baye’s Theorem)

**Definition 3 (Perfect Security)** A symmetric encryption is said to be perfectly secure if \( \forall m_0, m_1 \in M, c \in C \), and for all probability distributions over $M$,

$$Pr(C = c|M = m_0) = Pr(C = c|M = m_1)$$

Intuitively, this means that all messages are equally likely to be the plaintext corresponding to a given ciphertext. In other words, it is impossible to distinguish between the distinguish between the ciphertext corresponding to any two messages.

The two definitions can be shown to be equivalent. In other words, Definition3 $\Leftrightarrow$ Definition2. Thus, all three definitions are equivalent.
4.1 Construction of a Perfectely Secure SE: One Time Pad (OTP)

One Time Pad (OTP) encryption scheme provides perfect security. The message space, key space and ciphertext space in OTP are all of same size. OTP consists of:

- **Key Generation**: A randomized algorithm that outputs a random key.
  - Input: Null
  - $k \leftarrow \text{KeyGen}()$ (a random key picked from $\{0, 1\}^l$)

- **Encryption**: The cipher text 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 cipher text $c$ with key $k$ i.e.
  $$m = c \oplus k$$

**Theorem 1**  *One Time Pad is perfectly secure*

**Proof.** We use Definition 3 to prove our theorem.

$$\forall m_0, m_1 \in \mathcal{M}, \forall c \in \mathcal{C} \text{ and for all distributions over } \mathcal{M}, \text{ we have the following:}$$

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

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

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

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

Similarly,

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

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

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

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

This implies that:

$$Pr(C = c | M = m_0) = Pr(C = c | M = m_1)$$
Thus, by Definition 3 (for perfect security), OTO is a perfectly secure SE.

**Example to illustrate perfect secrecy of OTP:** Let \( l = 2 \).

Given that the ciphertext \( c = '10' \) Since, \( c = m \oplus k \)

- \( m = c \oplus k \)
- \( 10 = 10 \oplus 00 \)
- \( 00 = 10 \oplus 10 \)
- \( 11 = 10 \oplus 01 \)
- \( 01 = 10 \oplus 11 \)

\[
Pr(Enc(m, k) = c) = \frac{1}{2^2} = \frac{1}{4}
\]

It can be seen from the above example that a single cipher text ‘10’ is generated for 4 different possible combinations of \( k \) and \( m \). So the probability of figuring out the message is \( \frac{1}{4} \).

**Theorem 2** (Impracticality of Perfectly Secure Encryption) If there is a perfectly secure scheme, then 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 \( M \). Therefore,

\[
Pr(M = m_i | C = c) = 0 \quad \text{for some } m_i \in M
\]

However, for the remaining messages \( m_j \) such that \( m_j = Dec(c, k_j) \), 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 1) and thus leads to a contradiction.

**A note on the (im)practicality of perfectly secure encryption:** 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.