We are not going to formally define a notion of multiple-message secrecy
• Instead, define something stronger: security against chosen-plaintext attacks (CPA-security)
• minimal notion of security an encryption scheme should satisfy
Security against Chosen Plaintext Attack: Impossible?

It really is a problem if an attacker can tell when the same message is encrypted twice!

*This attack only works if encryption is deterministic!*
Random Functions

• Functions map from some set X to a set $F(X) = Y$.
  • (think of this mapping as a hash table mapping from $x \rightarrow y$)

• $\text{Func}_n$: all mappings from $X: \{0, 1\}^n \rightarrow F(X) = Y: \{0, 1\}^n$
  • i.e., for all input bit strings of length $n$, there is a mapping
to an output bit string also of length $n$
  • all possible mappings? $2^n \cdot (2^n)!!$ astronomically large!
Random Functions

Out of all possible functions between X and Y we choose one uniformly at random.

- e.g. for a 2 bit string mappings between X: \(\{0, 1\}^2\) and Y: \(\{0, 1\}^2\)
- one possible mapping that we could choose:

<table>
<thead>
<tr>
<th>x</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>F(x)</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>10</td>
</tr>
</tbody>
</table>

Properties of function \(F(X)\) chosen uniformly at random:

- for any given \(x \in X\), the probability that \(F(x) = y\) is \(1/2^n\)
- in our example example:
  - given \(x \in X\), the probability that \(F(x) = 1/2^2 = 1/4 = 0.25\)
- \(F(x)\) property:
  - if \(x\) changes by one bit to give \(x'\) then
    - \(F(x')\) is completely independent of \(F(x)\).
Random Permutations

• Variant of random function is random permutation
  • treat them equivalently for our purposes.

• E.g.: random permutation over bit strings of length 2
  Encryption: \( \{0, 1\}^2 \rightarrow \{0, 1\}^2 \)

Important Property of the Random Permutation:
A permutation is invertible (bijective) function

Given \( F(x) \) it is impossible to determine \( x \) without resorting to a brute force attack.

If \( |X| \) is very large? brute force not possible by an efficient (probabilistic polynomial time) attacker.
What we have, ideally: Random Functions

Consider the set of all permutations $F_k : X \rightarrow X$

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>01234...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_2$</td>
<td>10234...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{</td>
<td>X</td>
</tr>
</tbody>
</table>

Think of $X$ as all 128-bit bit strings

- If you know $k$, then $F_k(x)$ is trivial to invert
- If you don’t know $k$, then $F_k(x)$ is one-way
- One-way trapdoor function
What we have, ideally: Random Functions

Consider the set of all permutations $F_k: X \to X$

Think of $X$ as all 128-bit bit strings

Without knowing $k$, Eve learns nothing about $m$

$k$ is our key!
What we have, ideally: Random Functions

In essence, this protocol is saying “Let’s use the $i$th permutation function”
Infeasible to store all permutation functions – so instead cryptographers construct pseudorandom functions

Shared secret: index $k$ chosen uniformly, at random

Without knowing $k$, Eve learns nothing about $m$
What we have, approximately: Pseudo-Random Functions

In essence, this protocol is saying “Let’s use the $i$th permutation function”
Infeasible to store all permutation functions – so instead cryptographers construct pseudorandom functions

Shared secret: index $k$ chosen uniformly, at random

Without knowing $k$, Eve learns nothing about $m$
A Perfectly Secure Encryption Scheme

Regardless of any prior information the attacker has about the plaintext, the ciphertext observed by the attacker should leak no additional information about the plaintext.

Alice can only observe one ciphertext going over the network
Computational Secrecy

Would be okay if a scheme leaked information with a tiny probability to eavesdroppers with bounded computational resources.

• Allowing security to fail with a tiny probability (negligible in key length $n$)
  • how tiny is tiny? $2^{-60}$: probability of an event occurring every 100 billion years!

• Only consider efficient attackers (bounded in polynomial time by key length)
  • attackers that can brute-force the key space in bounded time.
  • try testing $2^{112}$ keys? Would take a supercomputer since Big Bang!
  • modern key space? $2^{128}$ or more!
Multiple message secrecy: Impossible?

It really is a problem if an attacker can tell when the same message is encrypted twice!

This attack only works if encryption is deterministic!
Random Permutations

• Variant of random function is random permutation
  • treat them equivalently for our purposes.

• E.g.: random permutation over bit strings of length 2
  Encryption: \( \{0, 1\}^2 \rightarrow \{0, 1\}^2 \)

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

Important Property of the Random Permutation:

A permutation is invertible (bijective) function

Given \( F(x) \) it is impossible to determine \( x \) without resorting to a brute force attack.

If \( |X| \) is very large? brute force not possible by an efficient (probabilistic polynomial time) attacker.
What we have, ideally: Random Functions

Consider the set of all permutations $F_k: X \rightarrow X$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$F(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

Shared secret: index $k$ chosen uniformly, at random

If you know $k$, then $F_k(x)$ is trivial to invert

If you don't know $k$, then $F_k(x)$ is one-way

Without knowing $k$, Eve learns nothing about $m$

$k$ is our key!
BLACKBOXES To this end, we’ll cover several “blackboxes”: what properties do they provide, and how can we responsibly put them together

Block ciphers  MACs  Hash functions  Public key crypto
Scenarios and Goals

Confidentiality
Keep others from reading Alice’s messages/data

Integrity
Keep others from undetectably tampering with Alice’s messages/data

Authenticity
Keep others from undetectably impersonating Alice (keep her to her word too!)

Block Ciphers
Block Ciphers

**ENCRYPTION**

Encryption Function: $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$

Fix the key $K$, then, $E_K: \{0, 1\}^n \rightarrow \{0, 1\}^n$

- plaintext size: $n$
- ciphertext size: $n$

$E_K$: permutation on $n$-bit strings.

- invertible (bijective function) given the key

Once the key is fixed: $E(k,m)$ is indistinguishable from a function chosen uniformly at random from all possible functions between block-sized binary strings.
Block Ciphers

**ENCRYPTION**

Once the key is fixed: \( E(k, m) \) is indistinguishable from a function chosen uniformly at random from all possible functions between block-sized binary strings.

Attacker has no way of knowing which random function was chosen to permute the plaintext to the ciphertext.
Inverse mapping of the permutation is the decryption algorithm, given the key
\[ D_k(E_k(M)) = M \]
without the key: best attack is a brute force exhaustive search over the entire key space!

Attacker has no way of knowing which random function was chosen to permute the plaintext to the ciphertext
## Block Ciphers

### Encryption

**Key** \( K \) $\rightarrow$ \( E \) $\rightarrow$ \( c \)

**AES key sizes:**
- 128, 192, 256

### Decryption

\( K \) $\rightarrow$ \( D \) $\rightarrow$ \( m \)

**Plaintext** ("message")

**Ciphertext**

**PROPERTY:**

- **Same fixed block size**
  - (AES: 128 bits, 3DES: 64 bits)

**PROPERTY:**

- **Small changes to the inputs cause big changes in the output**

**Confusion:** Each bit of the ciphertext should depend on each bit of the key

**Diffusion:** Flipping a bit in \( m \) should flip each bit in \( c \) with \( Pr = 1/2 \)
Encryption and Decryption and Key Generation Algorithm are publicly known. *The only unknown is the shared secret key*
Problem #1: Block Ciphers Are Deterministic

**PROPERTY:**
Block ciphers are deterministic
For a given \( m \) and \( K \),
\( E(K, m) \) always returns the same \( c \)

An eavesdropper could determine when messages are re-sent

**A FIX:**
Also known as an Initialization Vector or Nonce

\[ m \oplus r \text{ is the same size as } m \]

Choose random \( r \)
\[ K \rightarrow E \]
Send \( c \) and \( r \)
Choose random $r$ \hspace{1cm} K \rightarrow \hspace{1cm} E \hspace{1cm} \rightarrow \hspace{1cm} m \oplus r \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downarrow \hspace{1cm} \downar
Problem #2: Block Ciphers have fixed size

Fixed block size $m$

If we want to encrypt a message larger than the block size (128 bits), we simply break up the message into block-size length pieces...

...and encrypt each block

But recall: it can be deterministic. We must choose good initialization vectors. How?
Modes of Encryption: Electronic Codebook Mode (ECB)

**Encryption:**
inputs: plaintext: m, key: k,
ciphertext: c[i] = E(k, m[i])

**Decryption:**
inputs: ciphertext: c, key: k,
plaintext: m[i] = D(k, c[i])

Each block in AES 128 bits
ECB Mode

If two separate segments are equal, \( m[i] = m[j] \).
Then Eve can detect this by noting \( c[i] = c[j] \).

NEVER USE THE ECB MODE!

Same issue that led us to use Initialization Vectors!
NEVER use ECB  
(but over 50% of Android apps do)
Modes of Encryption: Cipher Block Chaining Mode (CBC)

Encryption
input: plaintext \( m \),
key \( k \),
initialization vector \( IV \)
\[
c[0] = IV \\
c[i] = E(k, m[i] \oplus c[i-1]) \text{ for } i \geq 1
\]

Decryption
input: ciphertext \( c \),
key \( k \),
initialization vector \( IV \)
\[
m[i] = D(k, c[i]) \oplus c[i-1]
\]
Modes of Encryption: Cipher Block Chaining Mode (CBC)

**Security**

*Input to the Encryption algorithm at each step is extremely likely to be different from the previous step.*

**Performance**

*Encryption: Not Parallelizable*

*Decryption: Parallelizable*

*recovering $m[i]$ does not require $m[i-1]$. Only requires $c[i-1]$ which is already known.*
Modes of Encryption: Cipher Feedback Mode (CFB)

**Encryption**
- Input: plaintext $m$, key $k$, initialization vector $IV$
  - $c[0] = IV$
  - $c[i] = E(k, c[i-1]) \oplus m[i]$ for $i \geq 1$

**Decryption**
- Input: ciphertext $c$, key $k$, initialization vector $IV$
  - $m[i] = E(k, c[i-1]) \oplus c[i]$

*Doesn't make use of the decryption function!*
Modes of Encryption: Cipher Feedback Mode (CFB)

Security:
\[ c[i] \neq c[j] \text{ for } m[i] = m[j] \]

Performance
* Encryption: Still Not Parallelizable
* Decryption: Parallelizable
  recovering \( m[i] \) does not require \( m[i-1] \). Only requires \( c[i-1] \) which is already known.

Doesn’t make use of the decryption function!
Original image

Encrypted using ECB mode

Modes other than ECB result in pseudo-randomness
Modes of Encryption: Counter Mode (CTR)

**Encryption**
input: plaintext m,
  key k,
  initialization vector IV
c[0] = IV
c[i] = E(k, IV+i) ⊕ m[i] for i >= 1

**Decryption**
input: ciphertext c,
  key k,
  initialization vector IV
m[i] = E(k, IV+i) ⊕ c[i]
Symmetric Key Cryptography

Confidentiality
- Keep others from reading Alice’s messages/data

Integrity
- Keep others from undetectably tampering with Alice’s messages/data

Authenticity
- Keep others from undetectably impersonating Alice (keep her to her word too!)

Block Ciphers

Limitations?
- what if Eve modifies the packet in transit?
- How do we share keys?
Scenarios and Goals

Confidentiality
Keep others from reading Alice’s messages/data

Integrity
Keep others from undetectably tampering with Alice’s messages/data

Message Authentication Codes (MACs)

Authenticity
Keep others from undetectably impersonating Alice (keep her to her word too!)
BLACKBOX #2:
MESSAGE AUTHENTICATION CODE (MAC)
Message Authentication Codes

**Confusion:** Each bit of the ciphertext should depend on each bit of the key

**Diffusion:** Flipping a bit in $m$ should flip each bit in $c$ with $Pr = 1/2$
Message Authentication Codes

- **Sign**: takes a key and a message and outputs a "tag"
  - $\text{Sgn}(k,m) = t$

- **Verify**: takes a key, a message, and a tag, and outputs Y/N
  - $\text{Vfy}(k,m,t) = \{Y,N\}$

- **Correctness**:
  - $\text{Vfy}(k, m, \text{Sgn}(k, m)) = Y$
Attacker Goal: Existential Forgery

- A MAC is secure if an attacker cannot demonstrate an existential forgery despite being able to perform a chosen plaintext attack:
  
  - Chose plaintext:
    - Attacker gets to choose m1, m2, m3, ...
    - And in return gets a properly computed t1, t2, t3, ...

  - Existential forgery:
    - Construct a new (m,t) pair such that Vfy(k, m, t) = Y
Encrypted CBC

Just take the last block in CBC  It’s a trap!

Cipher Block Chaining (CBC) mode encryption

Use a separate key and encrypt the last block
Hash Function Properties

- Very fast to compute
- Takes arbitrarily-sized inputs, returns fixed-sized output
- Pre-image resistant: Given $H(m)$, hard to determine $m$
- Collision resistant: Given $m$ and $H(m)$, hard to find $m' \neq m$ s.t. $H(m) = H(m')$

*Good hash functions: SHA family (SHA-256, SHA-512, …)*
Hash MACs

- **Sign(k, m):**
  - \( \text{opad} = 0x5c5c5c5c \ldots \)
  - \( \text{ipad} = 0x363636 \ldots \)
  - \( H((k \oplus \text{opad}) \| H((k \oplus \text{ipad}) \| m)) \)

- **Verify:**
  - Recompute and compare
CONFIDENTIALITY

Block ciphers

Deterministic ⇒ use IVs

Fixed block size ⇒ use encryption “modes”

INTEGRITY

Message Authentication Codes (MACs)

Send (message, tag) pairs

Verify that they match
CONFIDENTIALITY

Block ciphers

Deterministic $\Rightarrow$ use IVs

Fixed block size $\Rightarrow$ use encryption “modes”

INTEGRITY

Message Authentication Codes (MACs)

Send (message, tag) pairs

Verify that they match

Next

How do we establish $K$?

How do we know with whom we are communicating?