Encryption and Message Authentication
Bar-Ilan Winter School

Benny Applebaum
Tel-Aviv University

January, 2014

¹These slides are partially based on Benny Chor’s slides.
And Finally, Let’s Talk Business
And Finally, Let’s Talk Business

Encryption
Basic Setting

1. Eve listens to the communication.
2. Alice and Bob share a secret random key $k \in \{0, 1\}^n$.
3. Goal: Alice would like to send Bob a message $m$ confidentially.
Eve listens to the communication.
Basic Setting

1. Eve listens to the communication.
2. Alice and Bob share a secret random key $k \leftarrow \{0, 1\}^n$. 
Basic Setting

1. Eve listens to the communication.
2. Alice and Bob share a secret random key \( k \leftarrow \{0, 1\}^n \).
3. Goal: Alice would like to send Bob a message \( m \) confidentially.
Security Goals

There are some different goals we may be after
Security Goals

There are some different goals we may be after

- No adversary can learn $m$
Security Goals

There are some different goals we may be after

- No adversary can learn \( m \)
- No adversary can learn any meaningful information about \( m \).
Security Goals

There are some different goals we may be after

- No adversary can learn $m$
- No adversary can learn any meaningful information about $m$.
- No adversary can learn any information about $m$
Security Goals

There are some different goals we may be after

- No adversary can learn $m$
- No adversary can learn any meaningful information about $m$.
- No adversary can learn any information about $m$

Important questions:

- What are the adversary’s capabilities (e.g., passive/active) and knowledge (prior information)?
Security Goals

There are some different goals we may be after

- No adversary can learn $m$
- No adversary can learn any meaningful information about $m$.
- No adversary can learn any information about $m$

Important questions:

- What are the adversary’s capabilities (e.g., passive/active) and knowledge (prior information)?
- What are the adversary’s computational resources?
Security Goals

There are some different goals we may be after

- No adversary can learn $m$
- No adversary can learn any meaningful information about $m$.
- No adversary can learn any information about $m$

Important questions:

- What are the adversary’s capabilities (e.g., passive/active) and knowledge (prior information)?
- What are the adversary’s computational resources?
- Different answers lead to different security definitions.
Security Goals

There are some different goals we may be after

- No adversary can learn $m$
- No adversary can learn any meaningful information about $m$.
- No adversary can learn any information about $m$

Important questions:

- What are the adversary’s capabilities (e.g., passive/active) and knowledge (prior information)?
- What are the adversary’s computational resources?
- Different answers lead to different security definitions.
Security Goals

There are some different goals we may be after

- No adversary can learn $m$
- No adversary can learn any meaningful information about $m$. 
- No adversary can learn any information about $m$

Important questions:

- What are the adversary’s capabilities (e.g., passive/active) and knowledge (prior information)?
- What are the adversary’s computational resources?
- Different answers lead to different security definitions.

Meta question:

- Can we formalize secrecy mathematically?
Definition

A **symmetric encryption** scheme consists of:

- **Encryption Algorithm**: $E$ maps a key $k \in \{0, 1\}^*$ and a plaintext $m \in \{0, 1\}^*$ into a ciphertext $E_k(m)$. 

- **Decryption Algorithm**: $D$ maps a key $k \in \{0, 1\}^*$ and a ciphertext $c \in \{0, 1\}^*$ into a plaintext $D_k(c)$.
A symmetric encryption scheme consists of:

- **Encryption Algorithm**: $E$ maps a key $k \in \{0, 1\}^*$ and a plaintext $m \in \{0, 1\}^*$ into a ciphertext $E_k(m)$.

- **Decryption Algorithm**: $D$ maps a key $k \in \{0, 1\}^*$ and a ciphertext $c \in \{0, 1\}^*$ into a plaintext $D_k(c)$. 

The scheme should be correct:

\[ \forall m \in \{0, 1\}^*, k \in \{0, 1\}^* : D_k(E_k(m)) = m. \]
A symmetric encryption scheme consists of:

- **Encryption Algorithm**: $E$ maps a key $k \in \{0, 1\}^*$ and a plaintext $m \in \{0, 1\}^*$ into a ciphertext $E_k(m)$.

- **Decryption Algorithm**: $D$ maps a key $k \in \{0, 1\}^*$ and a ciphertext $c \in \{0, 1\}^*$ into a plaintext $D_k(c)$.
Encryption Syntax

Definition

A symmetric encryption scheme consists of:

- **Encryption Algorithm**: $E$ maps a key $k \in \{0, 1\}^*$ and a plaintext $m \in \{0, 1\}^*$ into a ciphertext $E_k(m)$.

- **Decryption Algorithm**: $D$ maps a key $k \in \{0, 1\}^*$ and a ciphertext $c \in \{0, 1\}^*$ into a plaintext $D_k(c)$.

The scheme should be correct:

$$\forall m \in \{0, 1\}^*, k \in \{0, 1\}^* : D_k(E_k(m)) = m.$$
Definition

A symmetric encryption scheme consists of:

- **Encryption Algorithm**: $E$ maps a key $k \in \{0, 1\}^*$ and a plaintext $m \in \{0, 1\}^*$ into a ciphertext $E_k(m)$.

- **Decryption Algorithm**: $D$ maps a key $k \in \{0, 1\}^*$ and a ciphertext $c \in \{0, 1\}^*$ into a plaintext $D_k(c)$.

The scheme should be correct:

$$\forall m \in \{0, 1\}^*, k \in \{0, 1\}^* : D_k(E_k(m)) = m.$$  

Note: Both algorithms are efficient and may be randomized.
A symmetric encryption scheme consists of:

- **Encryption Algorithm**: $E$ maps a key $k \in \{0, 1\}^*$ and a plaintext $m \in \{0, 1\}^*$ into a ciphertext $E_k(m)$.

- **Decryption Algorithm**: $D$ maps a key $k \in \{0, 1\}^*$ and a ciphertext $c \in \{0, 1\}^*$ into a plaintext $D_k(c)$.

The scheme should be correct:

$$\forall m \in \{0, 1\}^*, k \in \{0, 1\}^* : D_k(E_k(m)) = m.$$ 

Note: Both algorithms are efficient and may be randomized. So far, no requirement of secrecy.
Security as Indistinguishability

An encryption of $m_0$ and an encryption of $m_1$ should “look the same”.
Perfect Secrecy (Shannon ’49)

For any pair of different messages $m_0$ and $m_1$ of equal length: The ciphertexts $c_0$ and $c_1$ should be identically distributed.

**Experiment 0**

Let $k \leftarrow_R \{0, 1\}^n$

Output $c_0 = E_k(m_0)$

**Experiment 1**

Let $k \leftarrow_R \{0, 1\}^n$

Output $c_1 = E_k(m_1)$

- Very strong definition: can’t distinguish attack from retreat
Perfect Secrecy (Shannon ‘49)

For any pair of different messages $m_0$ and $m_1$ of equal length: The ciphertexts $c_0$ and $c_1$ should be identically distributed.

<table>
<thead>
<tr>
<th>Experiment 0</th>
<th>Experiment 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $k \xleftarrow{R} {0, 1}^n$</td>
<td></td>
</tr>
<tr>
<td>Output $c_0 = E_k(m_0)$</td>
<td></td>
</tr>
<tr>
<td>$\equiv$</td>
<td></td>
</tr>
<tr>
<td>Let $k \xleftarrow{R} {0, 1}^n$</td>
<td></td>
</tr>
<tr>
<td>Output $c_1 = E_k(m_1)$</td>
<td></td>
</tr>
</tbody>
</table>

- Very strong definition: can’t distinguish attack from retreat
- Example: one-time pad ($E_k(m) = k \oplus m$) is perfectly secret.
Perfect Secrecy (Shannon ’49)

For any pair of different messages $m_0$ and $m_1$ of equal length: The ciphertexts $c_0$ and $c_1$ should be identically distributed.

Experiment 0

Let $k \xleftarrow{R} \{0, 1\}^n$
Output $c_0 = E_k(m_0)$

Experiment 1

Let $k \xleftarrow{R} \{0, 1\}^n$
Output $c_1 = E_k(m_1)$

- Very strong definition: can’t distinguish attack from retreat
- Example: one-time pad ($E_k(m) = k \oplus m$) is perfectly secret.
- Unfortunately, perfect secrecy requires long key $|m| = |k|$
  (Ex: prove it!)
Computational Secrecy (Goldwasser & Micali ’82)

For any pair of different messages $m_0$ and $m_1$ of equal length:
The ciphertexts $c_0$ and $c_1$ should be indistinguishable for computationally-bounded adversary.

Experiment 0
Let $k \xleftarrow{R} \{0, 1\}^n$
Output $c_0 = E_k(m_0)$

Experiment 1
Let $k \xleftarrow{R} \{0, 1\}^n$
Output $c_1 = E_k(m_1)$
Computational Secrecy (Goldwasser & Micali ’82)

For any pair of different messages $m_0$ and $m_1$ of equal length: The ciphertexts $c_0$ and $c_1$ should be indistinguishable for computationally-bounded adversary.

**Experiment 0**

Let $k \xleftarrow{R} \{0, 1\}^n$
Output $c_0 = E_k(m_0)$

\[
\Pr[\mathcal{A}(c_0) = \text{accept}] - \Pr[\mathcal{A}(c_1) = \text{accept}] < \epsilon
\]

Outline: For any PPT adversary $\mathcal{A}$ and some negligible $\epsilon$.

**Experiment 1**

Let $k \xleftarrow{R} \{0, 1\}^n$
Output $c_1 = E_k(m_1)$
Computational Secrecy (Goldwasser & Micali ’82)

For any pair of different messages $m_0$ and $m_1$ of equal length:
The ciphertexts $c_0$ and $c_1$ should be indistinguishable for computationally-bounded adversary.

**Experiment 0**

Let $k \overset{R}{\leftarrow} \{0, 1\}^n$
Output $c_0 = E_k(m_0)$

**Experiment 1**

Let $k \overset{R}{\leftarrow} \{0, 1\}^n$
Output $c_1 = E_k(m_1)$

\[ \Pr[A(c_0) = \text{accept}] - \Pr[A(c_1) = \text{accept}] < \epsilon \]

For any PPT adversary $A$ and some negligible $\epsilon$. 
Computational Secrecy vs. Semantic Security

Comp. Secrecy is also known as Message Indistinguishability
Computational Secrecy vs. Semantic Security

Comp. Secrecy is also known as Message Indistinguishability

Semantic Security:

- “Everything that can be computed efficiently given the ciphertext can be also computed without the ciphertext”
Computational Secrecy vs. Semantic Security

Comp. Secrecy is also known as Message Indistinguishability

Semantic Security:

- “Everything that can be computed efficiently given the ciphertext can be also computed without the ciphertext”
- Therefore the ciphertext does not “add” useful information (for computationally bounded adversary)
Computational Secrecy vs. Semantic Security

Comp. Secrecy is also known as Message Indistinguishability

Semantic Security:

- “Everything that can be computed efficiently given the ciphertext can be also computed without the ciphertext”
- Therefore the ciphertext does not “add” useful information (for computationally bounded adversary)
- Exercise: try to formally define semantic security
Computational Secrecy vs. Semantic Security

Comp. Secrecy is also known as Message Indistinguishability

Semantic Security:

- “Everything that can be computed efficiently given the ciphertext can be also computed without the ciphertext”
- Therefore the ciphertext does not “add” useful information (for computationally bounded adversary)
- Exercise: try to formally define semantic security
Computational Secrecy vs. Semantic Security

Comp. Secrecy is also known as Message Indistinguishability

Semantic Security:

- “Everything that can be computed efficiently given the ciphertext can be also computed without the ciphertext”
- Therefore the ciphertext does not “add” useful information (for computationally bounded adversary)
- Exercise: try to formally define semantic security

Thm. Semantic Security is equivalent to Computational Secrecy (up to a polynomial loss in the parameters)
Computational Secrecy vs. Semantic Security

Comp. Secrecy is also known as Message Indistinguishability

Semantic Security:

- “Everything that can be computed efficiently given the ciphertext can be also computed without the ciphertext”
- Therefore the ciphertext does not “add” useful information (for computationally bounded adversary)
- Exercise: try to formally define semantic security

Thm. Semantic Security is equivalent to Computational Secrecy (up to a polynomial loss in the parameters)

Great! computational secrecy is a strong notion.
Is it feasible (with a short key)?
Computational analog of “one-time pad”

- Choose a secret random short key $k$ ("seed")
- Expand the seed into a long keying stream $G(k)$
- Encrypt $m$ by $c = G(k) \oplus m$
- Decrypt $c$ to $m = c \oplus G(k)$. 

![Diagram showing encryption and decryption process]
A pseudorandom generator is a polynomial time computable function $G : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$, $\ell \gg n$, which satisfies:

The output of $G$ is computationally indistinguishable from truly random strings of length $\ell$.
Theorem

Assume that $\text{PRG} : \{0,1\}^n \rightarrow \{0,1\}^\ell$ is pseudorandom. Then the “computational OTP” is secure.
From PRG to Encryption

**Theorem**

Assume that PRG : \{0, 1\}^n \rightarrow \{0, 1\}^\ell is pseudorandom.
Then the “computational OTP” is secure.

**Proof sketch.**

\[ E_k(m_0) \equiv (\text{PRG}(U_n) \oplus m_0) \overset{c}{=} (U_\ell \oplus m_0) \equiv U_\ell. \]
Theorem

Assume that PRG : \( \{0, 1\}^n \rightarrow \{0, 1\}^\ell \) is pseudorandom. Then the “computational OTP” is secure.

Proof sketch.

- \( E_k(m_0) \equiv (\text{PRG}(U_n) \oplus m_0) \overset{c}{=} (U_\ell \oplus m_0) \equiv U_\ell. \)

- For similar reason, \( E_k(m_1) \overset{c}{=} U_\ell. \)
Theorem

Assume that \( \text{PRG} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell \) is pseudorandom. Then the “computational OTP” is secure.

Proof sketch.

- \( E_k(m_0) \equiv (\text{PRG}(U_n) \oplus m_0) \equiv (U_\ell \oplus m_0) \equiv U_\ell \).
- For similar reason, \( E_k(m_1) \equiv U_\ell \).
- Hence, \( E_k(m_0) \equiv E_k(m_1) \).
Theorem

Assume that PRG : \{0, 1\}^n \rightarrow \{0, 1\}^\ell is pseudorandom. Then the “computational OTP” is secure.

Proof sketch.

- \( E_k(m_0) \equiv (\text{PRG}(U_n) \oplus m_0)^c \equiv (U_\ell \oplus m_0) \equiv U_\ell. \)
- For similar reason, \( E_k(m_1) \equiv U_\ell. \)
- Hence, \( E_k(m_0) \equiv E_k(m_1). \)
Multiple Messages

- We would like to use the same key to encrypt many messages
Multiple Messages

- We would like to use the same key to encrypt many messages.
- Recall that the PRG-based encryption is defined by
  \[ E_k(m) = \text{PRG}(k) \oplus m \]
Multiple Messages

- We would like to use the same key to encrypt many messages
- Recall that the PRG-based encryption is defined by
  \[ E_k(m) = \text{PRG}(k) \oplus m \]
- Is it ok to encrypt with the same key twice?
Multiple Messages

- We would like to use the same key to encrypt many messages
- Recall that the PRG-based encryption is defined by
  \[ E_k(m) = \text{PRG}(k) \oplus m \]
- Is it ok to encrypt with the same key twice?
- **Bad idea:** Given \( E_k(m_1) \) and \( E_k(m_2) \) the adversary learns whether \( m_1 = m_2 \) or more generally \( m_1 \oplus m_2 \)

Old versions of MS Word used an (excellent) PRG twice! As a result the encryption was completely broken and the plaintext was fully recovered! But we proved that the encryption is secure! What went wrong?
Multiple Messages

- We would like to use the same key to encrypt many messages.
- Recall that the PRG-based encryption is defined by $E_k(m) = \text{PRG}(k) \oplus m$.
- Is it ok to encrypt with the same key twice?
- **Bad idea**: Given $E_k(m_1)$ and $E_k(m_2)$ the adversary learns whether $m_1 = m_2$ or more generally $m_1 \oplus m_2$.
- Old versions of MS Word used an (excellent) PRG twice! As a result the encryption was completely broken and the plaintext was fully recovered!
Multiple Messages

- We would like to use the same key to encrypt many messages.
- Recall that the PRG-based encryption is defined by:
  \[ E_k(m) = \text{PRG}(k) \oplus m \]
- Is it ok to encrypt with the same key twice?
- **Bad idea:** Given \( E_k(m_1) \) and \( E_k(m_2) \) the adversary learns whether \( m_1 = m_2 \) or more generally \( m_1 \oplus m_2 \).
- Old versions of MS Word used an (excellent) PRG twice! As a result the encryption was completely broken and the plaintext was fully recovered!
- But we proved that the encryption is **secure**!
Multiple Messages

- We would like to use the same key to encrypt many messages.
- Recall that the PRG-based encryption is defined by
  \( E_k(m) = \text{PRG}(k) \oplus m \)
- Is it ok to encrypt with the same key twice?
- **Bad idea:** Given \( E_k(m_1) \) and \( E_k(m_2) \) the adversary learns whether \( m_1 = m_2 \) or more generally \( m_1 \oplus m_2 \)
- Old versions of MS Word used an (excellent) PRG twice! As a result the encryption was completely broken and the plaintext was fully recovered!
- But we proved that the encryption is **secure**!
- What went **wrong**?
Multiple Messages

- Our notion of security was defined for a single message
Our notion of security was defined for a single message. If we want to encrypt many messages we need a stronger definition.
Multiple Messages

- Our notion of security was defined for a single message
- If we want to encrypt many messages we need a stronger definition
- In fact, we would like to grant the adversary the extra power of Chosen Plaintext Attack
Multiple Messages

- Our notion of security was defined for a single message.
- If we want to encrypt many messages we need a stronger definition.
- In fact, we would like to grant the adversary the extra power of *Chosen Plaintext Attack*.
- Before that, let us reconsider our original definition.
Reminder: Ciphertext Indistinguishability

For any pair of messages $m_0$ and $m_1$ of equal length:

**Experiment 0**
Let $k \leftarrow \{0, 1\}^n$
Output $c_0 = E_k(m_0)$

**Experiment 1**
Let $k \leftarrow \{0, 1\}^n$
Output $c_1 = E_k(m_1)$
Ciphertext Indistinguishability: Alternative Formulation

Challenger

\[
k \overset{R}{\leftarrow} \{0, 1\}^n \\
b \overset{R}{\leftarrow} \{0, 1\}
\]

Adversary \( \mathcal{A}(1^n) \)

\[
\leftarrow (m_0, m_1) \\
E_k(m_b) \rightarrow
\]

Output \( b' \)

It is always possible to guess \( b \) with probability \( \frac{1}{2} \).

Security: For any PPT adversary \( \mathcal{A} \),

\[
\Pr[b' = b] \leq \frac{1}{2} + \text{neg}(n)
\]

Exercise: Prove equivalence to the original one.
Ciphertext Indistinguishability: Alternative Formulation

Challenger

\[ k \xleftarrow{R} \{0, 1\}^n \]
\[ b \xleftarrow{R} \{0, 1\} \]

\[ \leftarrow (m_0, m_1) \]
\[ E_k(m_b) \rightarrow \]

Adversary \( \mathcal{A}(1^n) \)

Output \( b' \)

- \( \mathcal{A} \) chooses a test \( m_0, m_1 \) and tries to distinguish \( E_k(m_0) \) from \( E_k(m_1) \)
Ciphertext Indistinguishability: Alternative Formulation

**Challenger**

\[ k \overset{R}{\leftarrow} \{0, 1\}^n \]
\[ b \overset{R}{\leftarrow} \{0, 1\} \]

\[ E_k(m_b) \rightarrow \]

**Adversary \( \mathcal{A}(1^n) \)**

\[ \leftarrow (m_0, m_1) \]

- \( \mathcal{A} \) chooses a test \( m_0, m_1 \) and tries to distinguish \( E_k(m_0) \) from \( E_k(m_1) \)

- It is always possible to guess \( b \) with probability \( \frac{1}{2} \)

\[ \text{Output } b' \]
Ciphertext Indistinguishability: Alternative Formulation

**Challenger**

\[ k \xleftarrow{R} \{0, 1\}^n \]
\[ b \xleftarrow{R} \{0, 1\} \]

\[ \leftarrow (m_0, m_1) \]
\[ E_k(m_b) \to \]

**Adversary \( \mathcal{A}(1^n) \)**

Output \( b' \)

- \( \mathcal{A} \) chooses a test \( m_0, m_1 \) and tries to distinguish \( E_k(m_0) \) from \( E_k(m_1) \)
- It is always possible to guess \( b \) with probability \( \frac{1}{2} \)
- Security: For any PPT adversary \( \mathcal{A} \),

\[ \Pr[b' = b] \leq \frac{1}{2} + \text{neg}(n) \]
Ciphertext Indistinguishability: Alternative Formulation

Challenger

\[
\begin{align*}
  k &\xleftarrow{\$} \{0, 1\}^n \\
  b &\xleftarrow{\$} \{0, 1\}
\end{align*}
\]

\[
E_k(m_b) \rightarrow
\]

Adversary \( \mathcal{A}(1^n) \)

Output \( b' \)

- \( \mathcal{A} \) chooses a test \( m_0, m_1 \) and tries to distinguish \( E_k(m_0) \) from \( E_k(m_1) \)
- It is always possible to guess \( b \) with probability \( \frac{1}{2} \)
- Security: For any PPT adversary \( \mathcal{A} \),

\[
\Pr[b' = b] \leq \frac{1}{2} + \text{neg}(n)
\]

- Exercise: Prove equivalence to the original one.
Indistinguishability under Chosen Plaintext Attack

**Challenger**

\[ k \xleftarrow{R} \{0, 1\}^n \]

\[ b \xleftarrow{R} \{0, 1\} \]

**Adversary** \( A(1^n) \)

1. \( x_1 \xleftarrow{} \)
2. \( E_k(x_1) \rightarrow \)
3. \( x_2 \xleftarrow{} \)
4. \( E_k(x_2) \rightarrow \)
5. \( \ldots \)

\[ \leftarrow (m_0, m_1) \]

\[ E_k(m_b) \rightarrow \]

**Output** \( b' \)
Indistinguishability under Chosen Plaintext Attack

The game has two phases:

**Challenger**

\[ k \xleftarrow{R} \{0, 1\}^n \]

\[ b \xleftarrow{R} \{0, 1\} \]

**Adversary** $\mathcal{A}(1^n)$

\[ \leftarrow x_1 \]

\[ E_k(x_1) \rightarrow \]

\[ \leftarrow x_2 \]

\[ E_k(x_2) \rightarrow \]

\[ \ldots \]

\[ \leftarrow (m_0, m_1) \]

\[ E_k(m_b) \rightarrow \]

Output $b'$
The game has two phases:

1. $\mathcal{A}$ is allowed to adaptively choose many encryptions
Indistinguishability under Chosen Plaintext Attack

The game has two phases:

1. $\mathcal{A}$ is allowed to adaptively choose many encryptions
2. $\mathcal{A}$ chooses a test $m_0, m_1$ and tries to distinguish $E_k(m_0)$ from $E_k(m_1)$
Chosen Plaintext Security

**Challenger**

\[ k \leftarrow R \{0, 1\}^n \]

\[ b \leftarrow R \{0, 1\} \]

**Adversary \( A(1^n) \)**

\[ x_1 \]

\[ E_k(x_1) \rightarrow \]

\[ x_2 \]

\[ E_k(x_2) \rightarrow \]

\[ \ldots \]

\[ (m_0, m_1) \]

\[ E_k(m_b) \rightarrow \]

Output \( b' \)

Security: For every PPT adversary

\[ \Pr[b = b'] \leq \frac{1}{2} + \text{neg}(n) \]

It is always possible to guess \( b \) with probability \( \frac{1}{2} \)

The adversary cannot do much better!
Chosen Plaintext Security

**Challenger**

\[ k \overset{R}{\leftarrow} \{0, 1\}^n \]

\[ b \overset{R}{\leftarrow} \{0, 1\} \]

**Adversary** \( A(1^n) \)

\[ \leftarrow x_1 \]
\[ E_k(x_1) \rightarrow \]
\[ \leftarrow x_2 \]
\[ E_k(x_2) \rightarrow \]
\[ \ldots \]

\[ \leftarrow (m_0, m_1) \]
\[ E_k(m_b) \rightarrow \]

**Security:** For every PPT adversary \( \Pr[b = b'] \leq \frac{1}{2} + \text{neg}(n) \)
Chosen Plaintext Security

**Challenger**

\[ k \overset{R}{\leftarrow} \{0, 1\}^n \]

\[ b \overset{R}{\leftarrow} \{0, 1\} \]

**Adversary** \( A(1^n) \)

\[ \leftarrow x_1 \]

\[ E_k(x_1) \rightarrow \]

\[ \leftarrow x_2 \]

\[ E_k(x_2) \rightarrow \]

\[ \ldots \]

\[ \leftarrow (m_0, m_1) \]

\[ E_k(m_b) \rightarrow \]

**Security:** For every PPT adversary \( \Pr[b = b'] \leq \frac{1}{2} + \operatorname{neg}(n) \)

- It is always possible to guess \( b \) with probability \( \frac{1}{2} \)
**Chosen Plaintext Security**

**Challenger**

\[
k \overset{R}{\leftarrow} \{0, 1\}^n
\]

\[
b \overset{R}{\leftarrow} \{0, 1\}
\]

**Adversary** $A(1^n)$

\[
\begin{align*}
&\leftarrow x_1 \\
&E_k(x_1) \rightarrow \\
&\leftarrow x_2 \\
&E_k(x_2) \rightarrow \\
&\ldots \\
&\leftarrow (m_0, m_1) \\
&E_k(m_b) \rightarrow
\end{align*}
\]

**Security:** For every PPT adversary $\Pr[b = b'] \leq \frac{1}{2} + \text{neg}(n)$

- It is always possible to guess $b$ with probability $\frac{1}{2}$
- The adversary cannot do much better!
Why do we need such a strong definition?

- Is it reasonable to assume that the adversary has an access to an Encryption Oracle?
Why do we need such a strong definition?

- Is it reasonable to assume that the adversary has an access to an Encryption Oracle?
- History: Yes!
Why do we need such a strong definition?

- Is it reasonable to assume that the adversary has an access to an Encryption Oracle?
- History: Yes!
- Example: Servers may communicate via encryption but (dishonest) users can control the actual requests that are being transferred
Why do we need such a strong definition?

- Is it reasonable to assume that the adversary has an access to an **Encryption Oracle**?
- **History:** Yes!
- **Example:** Servers may communicate via encryption but (dishonest) users can control the actual requests that are being transferred.
- **Remark:** One can define an intermediate notion (**Ciphertext Indistinguishability for Multiple Messages**) which is weaker than **CPA security** but stronger than **Ciphertext Indistinguishability for a single Message**.
Why do we need such a strong definition?

- Is it reasonable to assume that the adversary has an access to an Encryption Oracle?
- History: Yes!
- Example: Servers may communicate via encryption but (dishonest) users can control the actual requests that are being transferred
- Remark: One can define an intermediate notion (Ciphertext Indistinguishability for Multiple Messages) which is weaker than CPA security but stronger than Ciphertext Indistinguishability for a single Message.
- Ex: Try to formalize it and prove that it’s indeed strictly weaker than CPA and strictly stronger than CI for a single message.
Is CPA security realizable?

Theorem
If the encryption algorithm is a deterministic function $E_k(m)$ then it is insecure under chosen plaintext attacks (even if the adversary makes only one CPA query).

How can you prove it?

Does it mean that security under multiple messages cannot be achieved?

Q: How to bypass the limitation?

Sol1: Randomized encryption
Sol2: Stateful encryption
Is CPA security realizable?

Theorem

If the encryption algorithm is a deterministic function $E_k(m)$ then it is insecure under chosen plaintext attacks (even if the adversary makes only one CPA query).

How can you prove it?

Does it mean that security under multiple messages cannot be achieved?

Q: How to bypass the limitation?

Sol1: Randomized encryption

Sol2: Stateful encryption
Is CPA security realizable?

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the encryption algorithm is a deterministic function $E_k(m)$ then it is insecure under chosen plaintext attacks (even if the adversary makes only one CPA query).</td>
</tr>
</tbody>
</table>

How can you prove it?
Is CPA security realizable?

Theorem

If the encryption algorithm is a deterministic function $E_k(m)$ then it is insecure under chosen plaintext attacks (even if the adversary makes only one CPA query).

How can you prove it?

Does it mean that security under multiple messages cannot be achieved?
Is CPA security realizable?

Theorem

If the encryption algorithm is a deterministic function $E_k(m)$ then it is insecure under chosen plaintext attacks (even if the adversary makes only one CPA query).

How can you prove it?

Does it mean that security under multiple messages cannot be achieved?

Q: How to bypass the limitation?
Is CPA security realizable?

**Theorem**

If the encryption algorithm is a deterministic function $E_k(m)$ then it is **insecure** under chosen plaintext attacks (even if the adversary makes only one CPA query).

How can you prove it?

Does it mean that security under multiple messages **cannot** be achieved?

Q: How to bypass the limitation?

Sol1: Randomized encryption

Sol2: Stateful encryption
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

- For each input \( x \in \{0, 1\}^n \) choose \( R(x) \stackrel{R}{\leftarrow} \{0, 1\}^n \).
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

  - For each input \( x \in \{0, 1\}^n \) choose \( R(x) \overset{R}{\leftarrow} \{0, 1\}^n. \)

- How can we encrypt?
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

  - For each input \( x \in \{0, 1\}^n \) choose \( R(x) \leftarrow \{0, 1\}^n. \)

- How can we encrypt? Encrypt a message \( m \) by \( R(m) \).
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function 
\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

  - For each input \( x \in \{0, 1\}^n \) choose \( R(x) \leftarrow \{0, 1\}^n. \)

- How can we encrypt? Encrypt a message \( m \) by \( R(m) \).

- Decryption?
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

- For each input \( x \in \{0, 1\}^n \) choose \( R(x) \sim \{0, 1\}^n \).

- How can we encrypt? Encrypt a message \( m \) by \( R(m) \).

- Decryption?

- Let’s further assume that \( R \) is invertible, or even a permutation, hence \( R^{-1} : \{0, 1\}^n \rightarrow \{0, 1\}^n \) is used for decryption.
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

- For each input \( x \in \{0, 1\}^n \) choose \( R(x) \leftarrow \{0, 1\}^n \).

- How can we encrypt? Encrypt a message \( m \) by \( R(m) \).

- Decryption?

- Let’s further assume that \( R \) is invertible, or even a permutation, hence \( R^{-1} : \{0, 1\}^n \rightarrow \{0, 1\}^n \) is used for decryption.

- Security?
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

- For each input \( x \in \{0, 1\}^n \) choose \( R(x) \xleftarrow{R} \{0, 1\}^n. \)

- How can we encrypt? Encrypt a message \( m \) by \( R(m) \).

- Decryption?

- Let’s further assume that \( R \) is invertible, or even a permutation, hence \( R^{-1} : \{0, 1\}^n \rightarrow \{0, 1\}^n \) is used for decryption.

- Security?

- OK for (single-message) “Ciphertext Indistinguishability”
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

- For each input \( x \in \{0, 1\}^n \) choose \( R(x) \overset{R}{\leftarrow} \{0, 1\}^n. \)

- How can we encrypt? Encrypt a message \( m \) by \( R(m) \).

- Decryption?

- Let’s further assume that \( R \) is invertible, or even a permutation, hence \( R^{-1} : \{0, 1\}^n \rightarrow \{0, 1\}^n \) is used for decryption.

- Security?

- OK for (single-message) “Ciphertext Indistinguishability”

- How to achieve CPA security?
Encrypting via Ideal Cipher

- Suppose that Alice and Bob share a truly random function

\[ R : \{0, 1\}^n \rightarrow \{0, 1\}^n. \]

- For each input \( x \in \{0, 1\}^n \) choose \( R(x) \overset{R}{\leftarrow} \{0, 1\}^n. \)

- How can we encrypt? Encrypt a message \( m \) by \( R(m) \).

- Decryption?

- Let’s further assume that \( R \) is invertible, or even a permutation, hence \( R^{-1} : \{0, 1\}^n \rightarrow \{0, 1\}^n \) is used for decryption.

- Security?

- OK for (single-message) “Ciphertext Indistinguishability”

- How to achieve CPA security? Randomize the message!
(Inefficient) Construction

Encrypt \( m \): choose \( r \xleftarrow{R} \{0, 1\}^n \) and output \( (r, F(r \oplus m)) \)

Decrypt \( (r, c) \) compute \( r \oplus F^{-1}(c) \).
(Inefficient) Construction

Encrypt \( m \): choose \( r \xleftarrow{\$} \{0, 1\}^n \) and output \( (r, F(r \oplus m)) \)

Decrypt \( (r, c) \) compute \( r \oplus F^{-1}(c) \).

Theorem

*If \( F \) is random the scheme is CPA secure.*
(Inefficient) Construction

Encrypt $m$: choose $r \xleftarrow{R} \{0, 1\}^n$ and output $(r, F(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F^{-1}(c)$. 

Proof.

The adversary makes at most $t = \text{poly}(n)$ queries.

The $i$-th query $x_i$ is encrypted by $(r_i, c_i = F(x_i \oplus r_i))$.

The challenge $m_b$ is encrypted by $(r^*, c^* = F(m_b \oplus r^*))$.

Good event $G$:

$$\Pr_r \left[ G \right] \geq 1 - \frac{2t}{2n} = 1 - \text{neg}(n).$$

If $G$ happens, then conditioned on all seen ciphertexts, $(r^*, F(m_0 \oplus r^*)) \equiv (r^*, F(m_1 \oplus r^*))$.

Overall, the winning probability is upper-bounded by $\Pr[\text{win} | G] \Pr[G] + \Pr[\neg G] \leq \frac{1}{2} + \text{neg}(n)$. 

Benny Applebaum (Tel-Aviv University)
CPA Security from Random Permutation

(Inefficient) Construction

Encrypt $m$: choose $r \xleftarrow{R} \{0, 1\}^n$ and output $(r, F(r \oplus m))$
Decrypt $(r, c)$ compute $r \oplus F^{-1}(c)$.

Proof.

The adversary makes at most $t = \text{poly}(n)$ queries.
The $i$-th query $x_i$ is encrypted by $(r_i, c_i = F(x_i \oplus r_i))$.
The challenge $m_b$ is encrypted by $(r_*, c_* = F(m_b \oplus r_*))$. 
CPA Security from Random Permutation

(Inefficient) Construction

Encrypt $m$: choose $r \overset{R}{\leftarrow} \{0, 1\}^n$ and output $(r, F(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F^{-1}(c)$.

Proof.

The adversary makes at most $t = \text{poly}(n)$ queries.

The $i$-th query $x_i$ is encrypted by $(r_i, c_i = F(x_i \oplus r_i))$.

The challenge $m_b$ is encrypted by $(r_*, c_* = F(m_b \oplus r_*))$.

- **Good** event $G$: $(m_0 \oplus r_*)$ and $(m_1 \oplus r_*)$ not in $\{x_i \oplus r_i\}$
(Inefficient) Construction

Encrypt \( m \): choose \( r \overset{R}{\leftarrow} \{0, 1\}^n \) and output \((r, F(r \oplus m))\)  
Decrypt \((r, c)\) compute \( r \oplus F^{-1}(c)\).

Proof.

The adversary makes at most \( t = \text{poly}(n) \) queries.  
The \( i \)-th query \( x_i \) is encrypted by \((r_i, c_i = F(x_i \oplus r_i))\).  
The challenge \( m_b \) is encrypted by \((r_*, c_* = F(m_b \oplus r_*))\).

- **Good** event \( G \): \((m_0 \oplus r_*)\) and \((m_1 \oplus r_*)\) not in \( \{x_i \oplus r_i\}\).
- \( \Pr_{r_*}[G] \geq 1 - 2t/2^n = 1 - \text{neg}(n) \).
CPA Security from Random Permutation

(Inefficient) Construction

Encrypt $m$: choose $r \xleftarrow{R} \{0, 1\}^n$ and output $(r, F(r \oplus m))$
Decrypt $(r, c)$ compute $r \oplus F^{-1}(c)$.

Proof.

The adversary makes at most $t = \text{poly}(n)$ queries.
The $i$-th query $x_i$ is encrypted by $(r_i, c_i = F(x_i \oplus r_i))$.
The challenge $m_b$ is encrypted by $(r_*, c_* = F(m_b \oplus r_*))$.

- **Good** event $G$: $(m_0 \oplus r_*)$ and $(m_1 \oplus r_*)$ not in $\{x_i \oplus r_i\}$
- $\Pr_{r_*}[G] \geq 1 - 2t/2^n = 1 - \text{neg}(n)$.
- If $G$ happens, then conditioned on all seen ciphertexts,
  $(r_*, F(m_0 \oplus r_*)) \equiv (r_*, F(m_1 \oplus r_*))$. 
Encrypt $m$: choose $r \overset{R}{\leftarrow} \{0,1\}^n$ and output $(r, F(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F^{-1}(c)$.

Proof.

The adversary makes at most $t = \text{poly}(n)$ queries.

The $i$-th query $x_i$ is encrypted by $(r_i, c_i = F(x_i \oplus r_i))$.

The challenge $m_b$ is encrypted by $(r_*, c_* = F(m_b \oplus r_*))$.

- **Good** event $G$: $(m_0 \oplus r_*)$ and $(m_1 \oplus r_*)$ not in $\{x_i \oplus r_i\}$

  - $\Pr_{r_*}[G] \geq 1 - 2t/2^n = 1 - \text{neg}(n)$.

- If $G$ happens, then conditioned on all seen ciphertexts, $(r_*, F(m_0 \oplus r_*)) \equiv (r_*, F(m_1 \oplus r_*))$. 
(Inefficient) Construction

Encrypt $m$: choose $r \xleftarrow{R} \{0, 1\}^n$ and output $(r, F(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F^{-1}(c)$.

Proof.

The adversary makes at most $t = \text{poly}(n)$ queries.

The $i$-th query $x_i$ is encrypted by $(r_i, c_i = F(x_i \oplus r_i))$.

The challenge $m_b$ is encrypted by $(r_*, c_* = F(m_b \oplus r_*))$.

- **Good** event $G$: $(m_0 \oplus r_*)$ and $(m_1 \oplus r_*)$ not in $\{x_i \oplus r_i\}$
- $\Pr_{r_*}[G] \geq 1 - 2t/2^n = 1 - \text{neg}(n)$.

- If $G$ happens, then conditioned on all seen ciphertexts, $(r_*, F(m_0 \oplus r_*)) \equiv (r_*, F(m_1 \oplus r_*))$.

Overall, the winning probability is upper-bounded by

$\Pr[\text{win}|G] \Pr[G] + \Pr[\overline{G}] \leq \frac{1}{2} + \text{neg}(n)$. 

□
Pseudorandom Functions (Reminder)

Given a black-box access to the function, it’s infeasible to distinguish random function from pseudorandom function.
Pseudorandom Functions (Reminder)

Given a black-box access to the function, it’s infeasible to distinguish random function from pseudorandom function.

**PRF**

Let \( k \xleftarrow{R} \{0, 1\}^n \)

Given \( x \) output \( y = F_k(x) \)

**Random Function**

Choose random function \( R : \{0, 1\}^n \rightarrow \{0, 1\}^n \)

Given \( x \) output \( y = R(x) \)
Pseudorandom Functions (Reminder)

Given a black-box access to the function, it’s infeasible to distinguish random function from pseudorandom function.

**PRF**

Let $k \overset{R}{\leftarrow} \{0, 1\}^n$

Given $x$ output $y = F_k(x)$

**Random Function**

Choose random function $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$

Given $x$ output $y = R(x)$

PPT Adversary can’t distinguish with more than negligible probability.
Construction

Encrypt $m$: choose $r \leftarrow \{0, 1\}^n$ and output $(r, F_k(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F_k^{-1}(c)$.
Construction

Encrypt \( m \): choose \( r \leftarrow \{0, 1\}^n \) and output \((r, F_k(r \oplus m))\)

Decrypt \((r, c)\) compute \( r \oplus F_k^{-1}(c) \).

Theorem

If \( F \) is pseudorandom permutation the scheme is CPA secure.
 CPA Security from PRP (Proof)

Construction

Encrypt $m$: choose $r \overset{R}{\leftarrow} \{0, 1\}^n$ and output $(r, F_k(r \oplus m))$
Decrypt $(r, c)$ compute $r \oplus F_k^{-1}(c)$.

Proof by reduction: Convert a CPA attacker $A$ with success probability $\frac{1}{2} + \epsilon$ into an $\epsilon'$-distinguisher $B$ for the PRP.

\[ \text{Pr}_{k}[B \cdot F_k = 1] - \text{Pr}_{B \cdot \text{Rand}} = 1 \geq (\frac{1}{2} + \epsilon) - \text{neg}(n) \geq \epsilon - \text{neg}(n). \]
CPA Security from PRP (Proof)

**Construction**

Encrypt $m$: choose $r \leftarrow \{0, 1\}^n$ and output $(r, F_k(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F^{-1}_k(c)$.

**Proof by reduction:** Convert a CPA attacker $A$ with success probability $\frac{1}{2} + \epsilon$ into an $\epsilon'$-distinguisher $B$ for the PRP.

**Adversary $B^G$ ($G$ is either $F_k$ or Random)**

- Invoke $A$
CPA Security from PRP (Proof)

Construction

Encrypt $m$: choose $r \leftarrow \{0,1\}^n$ and output $(r, F_k(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F_k^{-1}(c)$.

Proof by reduction: Convert a CPA attacker $\mathcal{A}$ with success probability $\frac{1}{2} + \epsilon$ into an $\epsilon'$-distinguisher $\mathcal{B}$ for the PRP.

Adversary $\mathcal{B}^G$ ($G$ is either $F_k$ or Random))

- Invoke $\mathcal{A}$
- Answer a query $x_i$ with $(r_i \leftarrow \{0,1\}^n, G(r_i \oplus x_i))$. 

---

Benny Applebaum (Tel-Aviv University)  Encryption and Message Authentication  January, 2014  26 / 48
CPA Security from PRP (Proof)

**Construction**

Encrypt $m$: choose $r \overset{R}{\leftarrow} \{0, 1\}^n$ and output $(r, F_k(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F_k^{-1}(c)$.

**Proof by reduction:** Convert a CPA attacker $A$ with success probability $\frac{1}{2} + \epsilon$ into an $\epsilon'$-distinguisher $B$ for the PRP.

**Adversary $B^G$ ($G$ is either $F_k$ or Random)**

- Invoke $A$
- Answer a query $x_i$ with $(r_i \overset{R}{\leftarrow} \{0, 1\}^n, G(r_i \oplus x_i))$.
- Given $(m_0, m_1)$, send $(r^* \overset{R}{\leftarrow} \{0, 1\}^n, G(r^* \oplus m_b))$ where $b \overset{R}{\leftarrow} \{0, 1\}$.
CPA Security from PRP (Proof)

**Construction**

Encrypt \( m \): choose \( r \overset{R}{\leftarrow} \{0, 1\}^n \) and output \( (r, F_k(r \oplus m)) \).

Decrypt \((r, c)\) compute \( r \oplus F_k^{-1}(c) \).

**Proof by reduction:** Convert a CPA attacker \( \mathcal{A} \) with success probability \( \frac{1}{2} + \epsilon \) into an \( \epsilon' \)-distinguisher \( \mathcal{B} \) for the PRP.

**Adversary \( \mathcal{B}^G \) (\( G \) is either \( F_k \) or Random))**

- Invoke \( \mathcal{A} \)
- Answer a query \( x_i \) with \( (r_i \overset{R}{\leftarrow} \{0, 1\}^n, G(r_i \oplus x_i)) \).
- Given \( (m_0, m_1) \), send \( (r^* \overset{R}{\leftarrow} \{0, 1\}^n, G(r^* \oplus m_b)) \) where \( b \overset{R}{\leftarrow} \{0, 1\} \).
- Output 1 if \( \mathcal{A}'s \) guess \( b' \) equals to \( b \).
CPA Security from PRP (Proof)

Construction

Encrypt $m$: choose $r \leftarrow \{0, 1\}^n$ and output $(r, F_k(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F_k^{-1}(c)$.

Proof by reduction: Convert a CPA attacker $\mathcal{A}$ with success probability $\frac{1}{2} + \epsilon$ into an $\epsilon'$-distinguisher $\mathcal{B}$ for the PRP.

Adversary $\mathcal{B}^G$ ($G$ is either $F_k$ or Random))

- Invoke $\mathcal{A}$
- Answer a query $x_i$ with $(r_i \leftarrow \{0, 1\}^n, G(r_i \oplus x_i))$.
- Given $(m_0, m_1)$, send $(r^* \leftarrow \{0, 1\}^n, G(r^* \oplus m_b))$ where $b \leftarrow \{0, 1\}$.
- Output 1 if $\mathcal{A}$’s guess $b'$ equals to $b$. 
CPA Security from PRP (Proof)

Construction

Encrypt $m$: choose $r \leftarrow \{0, 1\}^n$ and output $(r, F_k(r \oplus m))$

Decrypt $(r, c)$ compute $r \oplus F^{-1}_k(c)$.

Proof by reduction: Convert a CPA attacker $A$ with success probability $\frac{1}{2} + \epsilon$ into an $\epsilon'$-distinguisher $B$ for the PRP.

Adversary $B^G$ ($G$ is either $F_k$ or Random))

- Invoke $A$
- Answer a query $x_i$ with $(r_i \leftarrow \{0, 1\}^n, G(r_i \oplus x_i))$.
- Given $(m_0, m_1)$, send $(r^* \leftarrow \{0, 1\}^n, G(r^* \oplus m_b))$ where $b \leftarrow \{0, 1\}$.
- Output 1 if $A$’s guess $b'$ equals to $b$.

$$\Pr_k[B^{F_k} = 1] - \Pr[B^{Rand} = 1] \geq \left(\frac{1}{2} + \epsilon\right) - \left(\frac{1}{2} + \text{neg}(n)\right) \geq \epsilon - \text{neg}(n).$$
CPA Security from Pseudorandom Function

Alternative Construction

Encrypt $m$: choose $r \xleftarrow{R} \{0, 1\}^n$ and output $(r, F_k(r) \oplus m)$
Encrypt $m$: choose $r \xleftarrow{R} \{0, 1\}^n$ and output $(r, F_k(r) \oplus m)$

Decrypt $(r, c)$ compute $F_k(r) \oplus c$. (No need to invert $F$)
CPA Security from Pseudorandom Function

**Alternative Construction**

Encrypt $m$: choose $r \xleftarrow{R} \{0, 1\}^n$ and output $(r, F_k(r) \oplus m)$  
Decrypt $(r, c)$ compute $F_k(r) \oplus c$. *(No need to invert $F$)*

Exercise prove:

**Theorem**

*If $F$ is pseudorandom function the scheme is CPA secure.*
How to encrypt long messages?

- Pseudorandom functions/permutations operate on blocks of fixed length (e.g., 128 bits).
How to encrypt long messages?

- Pseudorandom functions/permutations operate on blocks of fixed length (e.g., 128 bits).
- How to encrypt long messages?
How to encrypt long messages?

- Pseudorandom functions/permutations operate on blocks of fixed length (e.g., 128 bits).
- How to encrypt long messages?
- We can apply the previous constructions to each block separately but we’ll get poor rate (ciphertext is twice as large as the message)
How to encrypt long messages?

- Pseudorandom functions/permutations operate on blocks of fixed length (e.g., 128 bits).
- How to encrypt long messages?
- We can apply the previous constructions to each block separately but we’ll get poor rate (ciphertext is twice as large as the message)
- Is there a better solution?
CBC Mode Encryption

$E_k$ is a Pseudorandom permutation, $P_i$ is the $i$-th block of the message, and $S_0$ is a random seed (aka initialization vector (IV)).
CBC Mode Encryption

- $E_k$ is a Pseudorandom permutation, $P_i$ is the $i$-th block of the message, and $S_0$ is a random seed (aka initialization vector (IV)).
- The ciphertext is $(S_0, C_1, \ldots, C_n)$, the rate tends to 1 for long messages.
CBC Mode Encryption

- $E_k$ is a Pseudorandom permutation, $P_i$ is the $i$-th block of the message, and $S_0$ is a random seed (aka initialization vector (IV)).
- The ciphertext is $(S_0, C_1, \ldots, C_n)$, the rate tends to 1 for long messages.
- For a single block, we get the standard PRP-based Construction.
CBC Mode Encryption

- $E_k$ is a Pseudorandom permutation, $P_i$ is the $i$-th block of the message, and $S_0$ is a random seed (aka initialization vector (IV)).
- The ciphertext is $(S_0, C_1, \ldots, C_n)$, the rate tends to 1 for long messages.
- For a single block, we get the standard PRP-based Construction.
- New message requires a freshly chosen random seed (why?)
Properties of CBC

• Encryption seems inherently sequential – no parallel implementation known.
• Decryption is parallel – can decrypt the $i$-th block directly.
• Standard in most systems: SSL, IPSec, etc.

Security: It can be proved that if $E$ is a pseudorandom permutation, then CBC is resistant to chosen plaintext attacks (CPA-secure).
Properties of CBC

- Encryption seems inherently sequential – no parallel implementation known.
- Decryption is parallel – can decrypt the $i$-th block directly.
- Standard in most systems: SSL, IPSec, etc.

Security: It can be proved that if $E$ is a pseudorandom permutation, then CBC is resistant to chosen plaintext attacks (CPA-secure).
Properties of CBC

- Encryption seems inherently sequential – no parallel implementation known.
Properties of CBC

- Encryption seems inherently sequential – no parallel implementation known.
Properties of CBC

- Encryption seems inherently sequential – no parallel implementation known.
- Decryption is parallel – can decrypt the $i$-th block directly.
Properties of CBC

- Encryption seems inherently sequential – no parallel implementation known.
- Decryption is parallel – can decrypt the $i$-th block directly
Properties of CBC

- Encryption seems inherently sequential – no parallel implementation known.
- Decryption is parallel – can decrypt the $i$-th block directly
- **Standard** in most systems: SSL, IPSec, etc.
Properties of CBC

- Encryption seems inherently sequential – no parallel implementation known.
- Decryption is parallel – can decrypt the \( i \)-th block directly.
- **Standard** in most systems: SSL, IPSec, etc.

**Security:** It can be proved that if \( E \) is a pseudorandom permutation, then CBC is resistant to chosen plaintext attacks (CPA-secure).
Message Authentication Codes
Authentication – Goal

Ensure **integrity** of messages against an **active adversary**

- Adversary hears previous genuine messages
Authentication – Goal

Ensure **integrity** of messages against an **active adversary**

- Adversary hears previous genuine messages
- (May even influence the content of genuine messages)
Authentication – Goal

Ensure integrity of messages against an active adversary

- Adversary hears previous genuine messages
- (May even influence the content of genuine messages)
- Then sends own forged message(s).

Benny Applebaum (Tel-Aviv University)
Authentication – Goal

Ensure **integrity** of messages against an **active adversary**

- Adversary hears previous genuine messages
- (May even influence the content of genuine messages)
- Then sends own **forged message(s)**.
- Bob (receiver) should be able to tell genuine messages from forged ones.
Authentication – Goal

Ensure **integrity** of messages against an **active adversary**

- Adversary hears previous genuine messages
- (May even influence the content of genuine messages)
- Then sends own **forged message(s)**.
- Bob (receiver) should be able to tell genuine messages from forged ones.
Authentication – Goal

Ensure integrity of messages against an active adversary

- Adversary hears previous genuine messages
- (May even influence the content of genuine messages)
- Then sends own forged message(s).
- Bob (receiver) should be able to tell genuine messages from forged ones.

Important Remark: Authentication is orthogonal to secrecy. Secrecy alone usually does not guarantee integrity.
Sol: Message Authentication Code (MAC)

Idea: Alice and Bob share a secret key. Alice append to each message \( m \) an authentication tag \( \text{MAC}_k(m) = \text{tag} \). Bob verifies authenticity by comparing \( \text{MAC}_k(m) \) to \( \text{tag} \).
Sol: Message Authentication Code (MAC)

Idea: Alice and Bob share a secret key. Alice append to each message \( m \) an authentication tag \( \text{MAC}_k(m) = \text{tag} \). Bob verifies authenticity by comparing \( \text{MAC}_k(m) \) to \( \text{tag} \).

Definition (Message Authentication Code)

- Message space \( \mathcal{M} \) (usually long binary strings, e.g., \( \{0, 1\}^* \))
Sol: Message Authentication Code (MAC)

Idea: Alice and Bob share a secret key. Alice append to each message $m$ an authentication tag $\text{MAC}_k(m) = \text{tag}$. Bob verifies authenticity by comparing $\text{MAC}_k(m)$ to tag.

Definition (Message Authentication Code)

- Message space $\mathcal{M}$ (usually long binary strings, e.g., $\{0, 1\}^*$)
- Secret authentication key – $k \in \{0, 1\}^n$
**Idea:** Alice and Bob share a secret key. Alice append to each message $m$ an authentication tag $\text{MAC}_k(m) = \text{tag}$. Bob verifies authenticity by comparing $\text{MAC}_k(m)$ to $\text{tag}$.

**Definition (Message Authentication Code)**

- Message space $\mathcal{M}$ (usually long binary strings, e.g., $\{0, 1\}^*$)
- Secret authentication key $- k \in \{0, 1\}^n$
- Authentication algorithm $- \text{MAC}_k(m) \mapsto \text{tag}$

Remark: the MAC function is not 1-to-1 (why?)

Security: Intuitively, should be hard to forge a valid tag even after seeing many legal tags.
Sol: Message Authentication Code (MAC)

Idea: Alice and Bob share a secret key. Alice append to each message \( m \) an authentication tag \( \text{MAC}_k(m) = \text{tag} \). Bob verifies authenticity by comparing \( \text{MAC}_k(m) \) to \( \text{tag} \).

Definition (Message Authentication Code)

- Message space \( \mathcal{M} \) (usually long binary strings, e.g., \( \{0, 1\}^* \))
- Secret authentication key \(- \ k \in \{0, 1\}^n \)
- Authentication algorithm \(- \ \text{MAC}_k(m) \mapsto \text{tag} \)
- Typically, \( \text{tag} \in \{0, 1\}^\ell \) where \( \ell \) is relatively short
Sol: Message Authentication Code (MAC)

Idea: Alice and Bob share a secret key. Alice append to each message \( m \) an authentication tag \( \text{MAC}_k(m) = \text{tag} \). Bob verifies authenticity by comparing \( \text{MAC}_k(m) \) to \( \text{tag} \).

**Definition (Message Authentication Code)**

- Message space \( \mathcal{M} \) (usually long binary strings, e.g., \( \{0, 1\}^* \))
- Secret authentication key – \( k \in \{0, 1\}^n \)
- Authentication algorithm – \( \text{MAC}_k(m) \mapsto \text{tag} \)
- Typically, \( \text{tag} \in \{0, 1\}^\ell \) where \( \ell \) is relatively short
**Sol: Message Authentication Code (MAC)**

**Idea:** Alice and Bob share a secret key. Alice append to each message $m$ an authentication tag $\text{MAC}_k(m) = \text{tag}$. Bob verifies authenticity by comparing $\text{MAC}_k(m)$ to $\text{tag}$.

**Definition (Message Authentication Code)**

- Message space $\mathcal{M}$ (usually long binary strings, e.g., $\{0, 1\}^*$)
- Secret authentication key $- k \in \{0, 1\}^n$
- Authentication algorithm $- \text{MAC}_k(m) \mapsto \text{tag}$
- Typically, $\text{tag} \in \{0, 1\}^\ell$ where $\ell$ is relatively short

**Remark:** the MAC function is not 1-to-1 (why?)
Idea: Alice and Bob share a secret key. Alice append to each message $m$ an authentication tag $\text{MAC}_k(m) = \text{tag}$. Bob verifies authenticity by comparing $\text{MAC}_k(m)$ to tag.

Definition (Message Authentication Code)
- Message space $\mathcal{M}$ (usually long binary strings, e.g., $\{0, 1\}^*$)
- Secret authentication key – $k \in \{0, 1\}^n$
- Authentication algorithm – $\text{MAC}_k(m) \mapsto \text{tag}$
- Typically, $\text{tag} \in \{0, 1\}^\ell$ where $\ell$ is relatively short

Remark: the MAC function is not 1-to-1 (why?)

Security: Intuitively, should be hard to forge a valid tag even after seeing many legal tags
Security

Definition (Existential Forgery under Chosen Plaintext Attack)

A MAC is secure if every PPT adversary $A$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ ($i = 1, 2, ..., t$), outputs a new valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.

- The probability is taken over the choice of a random key.
- Adversary can choose the messages.
- The adversary succeeds even if the message being forged is "meaningless". The reason is that it is hard to predict what has and what does not have a meaning in an unknown context, and how will Bob, the receiver, react to such successful forgery.
Definition (Existential Forgery under Chosen Plaintext Attack)

A MAC is secure if every PPT adversary $A$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.
Security

**Definition (Existential Forgery under Chosen Plaintext Attack)**

A MAC is secure if every PPT adversary $\mathcal{A}$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ $(i = 1, 2, \ldots, t)$, outputs a **new** valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.

- The probability is taken over the choice of a random key.
Security

**Definition (Existential Forgery under Chosen Plaintext Attack)**

A MAC is secure if every PPT adversary $\mathcal{A}$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.

- The probability is taken over the choice of a random key
- Adversary can choose the messages
Security

Definition (Existential Forgery under Chosen Plaintext Attack)

A MAC is secure if every PPT adversary $A$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.

- The probability is taken over the choice of a random key
- Adversary can choose the messages
A MAC is secure if every PPT adversary $A$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.

- The probability is taken over the choice of a random key
- Adversary can choose the messages
- The adversary succeeds even if the message being forged is “meaningless”. The reason is that it is hard to predict what has and what does not have a meaning in an unknown context, and how will Bob, the receiver, react to such successful forgery.
Definition (Existential Forgery under Chosen Plaintext Attack)

A MAC is secure if every PPT adversary $A$ which is allowed to ask for polynomially-many legal pairs $(m_i, MAC_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, MAC_k(m))$ with no more than negligible probability.
Trivial Attacks

Definition (Existential Forgery under Chosen Plaintext Attack)

A MAC is secure if every PPT adversary $A$ which is allowed to ask for polynomially-many legal pairs $(m_i, MAC_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, MAC_k(m))$ with no more than negligible probability.

- Guess the $\ell$-bit tag of a message $m$ – success probability $2^{-\ell}$.
Definition (Existential Forgery under Chosen Plaintext Attack)

A MAC is secure if every PPT adversary $\mathcal{A}$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.

- Guess the $\ell$-bit tag of a message $m$ – success probability $2^{-\ell}$.

Trivial Attacks
Trivial Attacks

Definition (Existential Forgery under Chosen Plaintext Attack)

A MAC is secure if every PPT adversary $\mathcal{A}$ which is allowed to ask for polynomially-many legal pairs $(m_i, \text{MAC}_k(m_i))$ ($i = 1, 2, \ldots, t$), outputs a new valid pair $(m, \text{MAC}_k(m))$ with no more than negligible probability.

- Guess the $\ell$-bit tag of a message $m$ – success probability $2^{-\ell}$.
- Guess the $n$-bit key and compute the tag a message $m$ – success probability $2^{-n}$.
Trivial Attacks

Definition (Existential Forgery under Chosen Plaintext Attack)
A MAC is secure if every PPT adversary \( A \) which is allowed to ask for polynomially-many legal pairs \((m_i, \text{MAC}_k(m_i)) \) \((i = 1, 2, \ldots, t)\), outputs a new valid pair \((m, \text{MAC}_k(m))\) with no more than negligible probability.

- Guess the \( \ell \)-bit tag of a message \( m \) – success probability \( 2^{-\ell} \).
- Guess the \( n \)-bit key and compute the tag a message \( m \) – success probability \( 2^{-n} \).
- Conclusion: key and tag should not be too short.
MACs for Short Messages

What would Shannon do?

Claim: If $MAC : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded).

Can you see why?

In a computational setting can use pseudorandom function

Theorem: A PRF is a secure MAC.

Proof idea: If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.

Problem: PRFs are defined for a fixed length (“block”), but we would like to support long messages!
MACs for Short Messages

- What would Shannon do?

Claim: If \( MAC : \{0, 1\}^n \rightarrow \{0, 1\}^\ell \) is a random function then it cannot be broken with probability better than \( 2^{-\ell} \) (even if the adversary is computationally unbounded).

Can you see why?

In a computational setting can use pseudorandom function

Theorem: A PRF is a secure MAC.

Proof idea: If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.

Problem: PRFs are defined for a fixed length (“block”), but we would like to support long messages!
MACs for Short Messages

- What would Shannon do?

- Claim: If MAC: \{0, 1\}^n \rightarrow \{0, 1\}^\ell is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded).

Can you see why?

- In a computational setting can use pseudorandom function
- Theorem: A PRF is a secure MAC.
  - Proof idea: If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.
  - Problem: PRFs are defined for a fixed length ("block"), but we would like to support long messages!
MACs for Short Messages

- What would Shannon do?
- **Claim**: If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded).
MACs for Short Messages

- What would Shannon do?
- **Claim:** If $\text{MAC} : \{0,1\}^n \rightarrow \{0,1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded).
MACs for Short Messages

- What would Shannon do?
- **Claim:** If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
MACs for Short Messages

- What would Shannon do?
- **Claim:** If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use **pseudorandom function**
MACs for Short Messages

- What would Shannon do?
- **Claim**: If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use **pseudorandom function**
MACs for Short Messages

- What would Shannon do?
- **Claim**: If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use pseudorandom function
- **Theorem**: A PRF is a secure MAC.
MACs for Short Messages

- What would Shannon do?
- **Claim:** If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use pseudorandom function
- **Theorem:** A PRF is a secure MAC.
- **Proof idea:** If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.
MACs for Short Messages

- What would Shannon do?
- **Claim**: If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use **pseudorandom function**
- **Theorem**: A PRF is a secure MAC.
- **Proof idea**: If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.
MACs for Short Messages

- What would Shannon do?
- **Claim:** If MAC: \( \{0, 1\}^n \rightarrow \{0, 1\}^\ell \) is a random function then it cannot be broken with probability better than \( 2^{-\ell} \) (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use pseudorandom function
- **Theorem:** A PRF is a secure MAC.
- **Proof idea:** If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.
- **Problem:** PRFs are defined for a fixed length ("block"), but we would like to support long messages!
MACs for Short Messages

- What would Shannon do?
- **Claim:** If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use **pseudorandom function**
- **Theorem:** A PRF is a secure MAC.
- **Proof idea:** If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.
- **Problem:** PRFs are defined for a fixed length ("block"), but we would like to support long messages!
MACs for Short Messages

- What would Shannon do?
- **Claim:** If $\text{MAC} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is a random function then it cannot be broken with probability better than $2^{-\ell}$ (even if the adversary is computationally unbounded). Can you see why?
- In a computational setting can use **pseudorandom function**
- **Theorem:** A PRF is a secure MAC.
- **Proof idea:** If the PRF was truly random function then hard to forge, hence an adversary that breaks the MAC allows to distinguish the PRF from truly random function.
- **Problem:** PRFs are defined for a fixed length ("block"), but we would like to support **long** messages!
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.

**Suggestions:** Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

1. $(F_k(M_1), \ldots, F_k(M_\ell))$
2. $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \leftarrow \{0, 1\}^n/3$
3. $(r, F_k(r, 1, M_1, \ell), \ldots, F_k(r, \ell, M_\ell, \ell))$, where $r \leftarrow \{0, 1\}^n/4$

**Thm:** (only) the last construction is secure!

**Ex:** Prove it.

**Problem:** Impractical due to large communication overhead!
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function. 

**Suggestions:** Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(r, F_k(r, M_1), \ldots, F_k(r, M_\ell))$, where $r \leftarrow \{0, 1\}^{n/3}$.
- $(r, F_k(r, M_1, \ell), \ldots, F_k(r, M_\ell, \ell))$, where $r \leftarrow \{0, 1\}^{n/4}$.
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function. 

Suggestions: Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$
- $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \xleftarrow{} \{0, 1\}^{n/3}$.
- $(r, F_k(r, 1, M_1, \ell), \ldots, F_k(r, \ell, M_\ell, \ell))$, where $r \xleftarrow{} \{0, 1\}^{n/4}$.

Thm: (only) the last construction is secure!

Ex: Prove it.

Problem: Impractical due to large communication overhead!
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.

**Suggestions:** Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$, where $r \xleftarrow{\$} \{0, 1\}^{n/3}$.
- $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \xleftarrow{\$} \{0, 1\}^{n/4}$.

**Thm:** (only) the last construction is secure!

**Ex:** Prove it.

**Problem:** Impractical due to large communication overhead!
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \to \{0, 1\}^n$ be a pseudorandom function. 

**Suggestions:** Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$.

**Theorem:** (only) the last construction is secure!

**Example:** Prove it.

**Problem:** Impractical due to large communication overhead!
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.  

**Suggestions:** Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$. 

Thm: (only) the last construction is secure! 
Ex: Prove it. 
Problem: Impractical due to large communication overhead!

Benny Applebaum (Tel-Aviv University)  
Encryption and Message Authentication  
January, 2014 37 / 48
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.

Suggestions: Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$.
- $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \leftarrow \{0, 1\}^{n/3}$. 

Thm: (only) the last construction is secure!

Ex: Prove it.

Problem: Impractical due to large communication overhead!
Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.

**Suggestions:** Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$.
- $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \leftarrow \{0, 1\}^{n/3}$.
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.

Suggestions: Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$.
- $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \leftarrow \{0, 1\}^{n/3}$.
- $(r, F_k(r, 1, M_1, \ell), \ldots, F_k(r, \ell, M_\ell), \ell)$, where $r \leftarrow \{0, 1\}^{n/4}$. 

Thm: (only) the last construction is secure!

Ex: Prove it.

Problem: Impractical due to large communication overhead!
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.

**Suggestions:** Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$.
- $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \leftarrow \{0, 1\}^{n/3}$.
- $(r, F_k(r, 1, M_1, \ell), \ldots, F_k(r, \ell, M_\ell), \ell)$, where $r \leftarrow \{0, 1\}^{n/4}$.

**Thm:** (only) the last construction is secure!

**Ex:** Prove it.
How to authenticate Long Messages?

Let $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a pseudorandom function.

Suggestions: Define $\text{MAC}_k(M_1, \ldots, M_\ell)$ as:

- $(F_k(M_1), \ldots, F_k(M_\ell))$
- $(F_k(1, M_1), \ldots, F_k(\ell, M_\ell))$.
- $(r, F_k(r, 1, M_1), \ldots, F_k(r, \ell, M_\ell))$, where $r \leftarrow \{0, 1\}^{n/3}$.
- $(r, F_k(r, 1, M_1, \ell), \ldots, F_k(r, \ell, M_\ell, \ell))$, where $r \leftarrow \{0, 1\}^{n/4}$.

Thm: (only) the last construction is secure!

Ex: Prove it.

Problem: Impractical due to large communication overhead!
MACs for Long Messages

We will describe an efficient approach based on CBC Mode, there is an alternative solution (HMAC) based on cryptographic hash functions.
CBC Mode MACs

- Start with the all zero seed.

```
\begin{align*}
00000000 & \rightarrow \text{M}_1 \\
\text{E}_k & \rightarrow \text{C}_1 \\
\text{M}_2 & \rightarrow \text{E}_k \\
\text{C}_2 & \rightarrow \text{E}_k \\
\text{M}_n & \rightarrow \text{E}_k \\
\text{C}_n & \rightarrow \text{E}_k
\end{align*}
```
CBC Mode MACs

- Start with the all zero seed.
- Given a message consisting of $n$ blocks, $M_1, M_2, \ldots, M_n$, apply CBC mode encryption (using the secret key $k$).

Q: Can we replace the all-zero seed with a random public string?
CBC Mode MACs

- Start with the all zero seed.
- Given a message consisting of \( n \) blocks, \( M_1, M_2, \ldots, M_n \), apply CBC mode encryption (using the secret key \( k \)).

- Produce \( n \) “cipher text” blocks, \( C_1, C_2, \ldots, C_n \).
CBC Mode MACs

- Start with the **all zero** seed.
- Given a message consisting of $n$ blocks, $M_1, M_2, \ldots, M_n$, apply CBC mode encryption (using the secret key $k$).

- Produce $n$ “ciphertext” blocks, $C_1, C_2, \ldots, C_n$.
- Discard first $n - 1$ blocks.
CBC Mode MACs

- Start with the all zero seed.
- Given a message consisting of \( n \) blocks, \( M_1, M_2, \ldots, M_n \), apply CBC mode encryption (using the secret key \( k \)).

- Produce \( n \) “cipertext” blocks, \( C_1, C_2, \ldots, C_n \).
- Discard first \( n - 1 \) blocks.
- Send \( M_1, M_2, \ldots, M_n \) and the tag \( \text{MAC}_k(M) = C_n \).
CBC Mode MACs

- Start with the all zero seed.
- Given a message consisting of \( n \) blocks, \( M_1, M_2, \ldots, M_n \), apply CBC mode encryption (using the secret key \( k \)).

- Produce \( n \) “cipher text” blocks, \( C_1, C_2, \ldots, C_n \).
- Discard first \( n - 1 \) blocks.
- Send \( M_1, M_2, \ldots, M_n \) and the tag \( \text{MAC}_k(M) = C_n \).
CBC Mode MACs

- Start with the \textbf{all zero} seed.
- Given a message consisting of \( n \) blocks, \( M_1, M_2, \ldots, M_n \), apply CBC mode \textit{encryption} (using the secret key \( k \)).

![Diagram of CBC Mode MACs]

- Produce \( n \) “cipertext” blocks, \( C_1, C_2, \ldots, C_n \).
- Discard first \( n - 1 \) blocks.
- Send \( M_1, M_2, \ldots, M_n \) and the tag \( \text{MAC}_k(M) = C_n \).

Q: Can we replace the all-zero seed with a random public string?
Theorem: If $E_k$ is a pseudorandom function, then the fixed length CBC MAC is resilient to forgery when authenticating messages of the same length, $n$. 

Warning: Construction is not secure if messages are of varying lengths, namely number of blocks varies among messages.
Theorem: If $E_k$ is a pseudorandom function, then the fixed length CBC MAC is resilient to forgery when authenticating messages of the same length, $n$. 

Warning: Construction is not secure if messages are of varying length, namely number of blocks varies among messages.
Security of Fixed Length CBC MAC [BKR, 2000]

- **Theorem:** If $E_k$ is a pseudorandom function, then the fixed length CBC MAC is resilient to forgery when authenticating messages of the same length, $n$. 

- **Proof via reduction:** Assume CBC MAC can be forged efficiently. Transform the forging algorithm into an algorithm distinguishing $E_k$ from a random function efficiently.

- **Warning:** Construction is not secure if messages are of varying length, namely number of blocks varies among messages.
Security of Fixed Length CBC MAC [BKR, 2000]

- **Theorem:** If $E_k$ is a pseudorandom function, then the fixed length CBC MAC is resilient to forgery when authenticating messages of the same length, $n$.

- **Proof via reduction:** Assume CBC MAC can be forged efficiently. Transform the forging algorithm into an algorithm distinguishing $E_k$ from a random function efficiently.

Warning: Construction is not secure if messages are of varying lengths, namely number of blocks varies among messages.
Theorem: If $E_k$ is a pseudorandom function, then the fixed length CBC MAC is resilient to forgery when authenticating messages of the same length, $n$.

Proof via reduction: Assume CBC MAC can be forged efficiently. Transform the forging algorithm into an algorithm distinguishing $E_k$ from a random function efficiently.
Theorem: If $E_k$ is a pseudorandom function, then the fixed length CBC MAC is resilient to forgery when authenticating messages of the same length, $n$.

Proof via reduction: Assume CBC MAC can be forged efficiently. Transform the forging algorithm into an algorithm distinguishing $E_k$ from a random function efficiently.
Security of Fixed Length CBC MAC [BKR, 2000]

- **Theorem:** If $E_k$ is a pseudorandom function, then the fixed length CBC MAC is resilient to forgery when authenticating messages of the same length, $n$.

- **Proof via reduction:** Assume CBC MAC can be forged efficiently. Transform the forging algorithm into an algorithm distinguishing $E_k$ from a random function efficiently.

- **Warning:** Construction is not secure if messages are of varying lengths, namely number of blocks varies among messages.
Insecurity of Variable Length CBC MAC

Here is a simple, chosen plaintext example of forgery:

- Get $C_1 = CBC - MAC_k(M_1) = E_k(\overrightarrow{0} \oplus M_1)$
Insecurity of Variable Length CBC MAC

Here is a simple, chosen plaintext example of forgery:

- Get $C_1 = CBC - MAC_k(M_1) = E_k(\vec{0} \oplus M_1)$
- Ask for MAC of $C_1$, i.e.,
  
  $C_2 = CBC - MAC_k(C_1) = E_k(\vec{0} \oplus C_1)$
Insecurity of Variable Length CBC MAC

Here is a simple, chosen plaintext example of forgery:

- Get $C_1 = CBC - MAC_k(M_1) = E_k(\vec{0} \oplus M_1)$
- Ask for MAC of $C_1$, i.e.,
  $C_2 = CBC - MAC_k(C_1) = E_k(\vec{0} \oplus C_1)$
- Observe that $E_k(C_1 \oplus \vec{0}) = E_k(E_k(\vec{0} \oplus M_1) \oplus \vec{0}) = 
  CBC - MAC_k(M_1 \circ \vec{0})$ (where $\circ$ denotes concatenation)
Insecurity of Variable Length CBC MAC

Here is a simple, chosen plaintext example of forgery:

- Get \( C_1 = CBC - MAC_k(M_1) = E_k(\vec{0} \oplus M_1) \)
- Ask for MAC of \( C_1 \), i.e.,
  \( C_2 = CBC - MAC_k(C_1) = E_k(\vec{0} \oplus C_1) \)
- Observe that \( E_k(C_1 \oplus \vec{0}) = E_k(E_k(\vec{0} \oplus M_1) \oplus \vec{0}) = CBC - MAC_k(M_1 \circ \vec{0}) \) (where \( \circ \) denotes concatenation)
Insecurity of Variable Length CBC MAC

Here is a simple, chosen plaintext example of forgery:

- Get $C_1 = CBC - MAC_k(M_1) = E_k(\vec{0} \oplus M_1)$
- Ask for MAC of $C_1$, i.e.,
  $C_2 = CBC - MAC_k(C_1) = E_k(\vec{0} \oplus C_1)$
- Observe that $E_k(C_1 \oplus \vec{0}) = E_k(E_k(\vec{0} \oplus M_1) \oplus \vec{0}) = CBC - MAC_k(M_1 \circ \vec{0})$ (where $\circ$ denotes concatenation)

- One can efficiently design, for every $n$, two messages, one with 1 block, the other with $n + 1$ blocks, that have the same $MAC_k(\cdot)$. 
CBC-MAC for Variable Length Messages

- **Solution 1:** The first block of the message is set to be its length. Apply CBC-MAC to $(n, M_1, \ldots, M_n)$. Works since now message space is prefix-free. Drawback: The message length, $n$, must be known in advance.
CBC-MAC for **Variable Length Messages**

- **Solution 1:** The first block of the message is set to be its length. Apply CBC-MAC to \((n, M_1, \ldots, M_n)\). Works since now message space is **prefix-free**. Drawback: The message length, \(n\), must be known in advance.

- **Solution 2:** Apply CBC-MAC to \((M_1, \ldots, M_n, n)\). Message length does not have to be known in advance. Looks good, but this scheme was broken.

- **Solution 3:** Encrypted CBC (ECBC MAC): Compute \(E_{k_2}(\text{CBC-MAC}_{k_1}(M_1, \ldots, M_n))\), where \(E\) is a block-cipher and \(k_2\) is another secret key. Essentially the same overhead as CBC-MAC (widely used).
CBC-MAC for **Variable Length Messages**

- **Solution 1**: The first block of the message is set to be its length. Apply CBC-MAC to \((n, M_1, \ldots, M_n)\).
  Works since now message space is **prefix-free**.
  Drawback: The message length, \(n\), must be known in advance.

- **“Solution 2”**: Apply CBC-MAC to \((M_1, \ldots, M_n, n)\).
  Message length **does not** have to be known in advance.
  Looks good, **but** this scheme was broken.
CBC-MAC for Variable Length Messages

- **Solution 1:** The first block of the message is set to be its length. Apply CBC-MAC to \((n, M_1, \ldots, M_n)\). Works since now message space is prefix-free. Drawback: The message length, \(n\), must be known in advance.

- **“Solution 2”:** Apply CBC-MAC to \((M_1, \ldots, M_n, n)\). Message length does not have to be known in advance. Looks good, but this scheme was broken.
CBC-MAC for Variable Length Messages

- **Solution 1**: The first block of the message is set to be its length. Apply CBC-MAC to \((n, M_1, \ldots, M_n)\). Works since now message space is prefix-free. Drawback: The message length, \(n\), must be known in advance.

- **“Solution 2”**: Apply CBC-MAC to \((M_1, \ldots, M_n, n)\). Message length does not have to be known in advance. Looks good, but this scheme was broken.

- **Solution 3**: Encrypted CBC (ECBC MAC): Compute \(E_{k_2}(CBC - MAC_{k_1}(M_1, \ldots, M_n))\), where \(E\) is a block-cipher and \(k_2\) is another secret key. Essentially the same overhead as CBC-MAC (widely used).
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate:**
  \[ E_{k_1}(M), \text{MAC}_{k_2}(M) \]
  Not secure (some MACs may leak information on \( M \))

- **Authenticate-then-Encrypt:**
  \[ E_{k_1}(M, \text{MAC}_{k_2}(M)) \]
  Not in general

- **Encrypt-then-Authenticate:**
  \[ E_{k_1}(M), \text{MAC}_{k_2}(E_{k_1}(M)) \]
  Secure
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?
Suggestions:

- Encrypt-and-Authenticate: $E_{k_1}(M), \text{MAC}_{k_2}(M)$
  - Secure? No (some MACs may leak information on $M$)

- Authenticate-then-Encrypt: $E_{k_1}(M, \text{MAC}_{k_2}(M))$
  - Secure? Not in general

- Encrypt-then-Authenticate: $E_{k_1}(M), \text{MAC}_{k_2}(E_{k_1}(M))$
  - Secure? Yes
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate:** $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate**: $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?

- **Authenticate-then-Encrypt**: $E_{k_1}(M, \text{MAC}_{k_2}(M))$ secure?

- **Encrypt-then-Authenticate**: $E_{k_1}(M), \text{MAC}_{k_2}(E_{k_1}(M))$ secure?
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?
Suggestions:

- **Encrypt-and-Authenticate**: $E_{k_1}(M)$, $\text{MAC}_{k_2}(M)$ secure?
  - No (some MACs may leak information on $M$)
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate**: $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?
  - **No** (some MACs may leak information on $M$)

- **Authenticate-then-Encrypt**: $E_{k_1}(M, \text{MAC}_{k_2}(M))$ secure?
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate**: $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?
  - No (some MACs may leak information on $M$)

- **Authenticate-then-Encrypt**: $E_{k_1}(M, \text{MAC}_{k_2}(M))$ secure?
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

• Encrypt-and-Authenticate: $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?
  No (some MACs may leak information on $M$)

• Authenticate-then-Encrypt: $E_{k_1}(M, \text{MAC}_{k_2}(M))$ secure?
  Not in general
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate:** $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?
  - No (some MACs may leak information on $M$)

- **Authenticate-then-Encrypt:** $E_{k_1}(M, \text{MAC}_{k_2}(M))$ secure?
  - Not in general

- **Encrypt-then-Authenticate:** $E_{k_1}(M), \text{MAC}_{k_2}(E_{k_1}(M))$ secure?
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate:** $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?
  - No (some MACs may leak information on $M$)

- **Authenticate-then-Encrypt:** $E_{k_1}(M, \text{MAC}_{k_2}(M))$ secure?
  - Not in general

- **Encrypt-then-Authenticate:** $E_{k_1}(M), \text{MAC}_{k_2}(E_{k_1}(M))$ secure?
Combining Authentication and Secrecy

It is a good idea to use two different keys: one for authentication and one for encryption. But How?

Suggestions:

- **Encrypt-and-Authenticate**: $E_{k_1}(M), \text{MAC}_{k_2}(M)$ secure?
  No (some MACs may leak information on $M$)

- **Authenticate-then-Encrypt**: $E_{k_1}(M, \text{MAC}_{k_2}(M))$ secure?
  Not in general

- **Encrypt-then-Authenticate**: $E_{k_1}(M), \text{MAC}_{k_2}(E_{k_1}(M))$ secure?
  Yes
Authenticated Encryption

Authentication is important even if one is interested only in secrecy!
Authenticated Encryption

Authentication is important even if one is interested only in secrecy!

Alice wants to send an $n$-bit message $M$ to Bob over a noisy channel. They share a secret-key of a CPA secure encryption $E_k$.

- Alice sends a bit-by-bit encryption $E_k(M_1), \ldots, E_k(M_n)$ together with an encryption of the parity-check $E_k(M_1 \oplus \ldots \oplus M_n)$ so that Bob can detect errors.
Authenticated Encryption

Authentication is important even if one is interested only in secrecy!

Alice wants to send an $n$-bit message $M$ to Bob over a noisy channel. They share a secret-key of a CPA secure encryption $E_k$.

- Alice sends a bit-by-bit encryption $E_k(M_1), \ldots, E_k(M_n)$ together with an encryption of the parity-check $E_k(M_1 \oplus \cdots \oplus M_n)$ so that Bob can detect errors.
- Bob decrypts. If the parity check does not match, he sends an error message.

How can an active adversary recover the message $M$?

CPA security is not always enough!

(Some real world attacks follow a similar scenario)
Authenticated Encryption

Authentication is important even if one is interested only in \textit{secrecy}!

Alice wants to send an $n$-bit message $M$ to Bob over a noisy channel. They share a secret-key of a CPA secure encryption $E_k$.

- Alice sends a bit-by-bit encryption $E_k(M_1), \ldots, E_k(M_n)$ together with an encryption of the parity-check $E_k(M_1 \oplus \ldots \oplus M_n)$ so that Bob can detect errors.
- Bob decrypts. If the parity check does not match, he sends an error message.
Authenticated Encryption

Authentication is important even if one is interested only in secrecy!

Alice wants to send an $n$-bit message $M$ to Bob over a noisy channel. They share a secret-key of a CPA secure encryption $E_k$.

- Alice sends a bit-by-bit encryption $E_k(M_1), \ldots, E_k(M_n)$ together with an encryption of the parity-check $E_k(M_1 \oplus \ldots \oplus M_n)$ so that Bob can detect errors.
- Bob decrypts. If the parity check does not match, he sends an error message.

How can an active adversary recover the message $M$?
Authenticated Encryption

Authentication is important even if one is interested only in secrecy!

Alice wants to send an $n$-bit message $M$ to Bob over a noisy channel. They share a secret-key of a CPA secure encryption $E_k$.

- Alice sends a bit-by-bit encryption $E_k(M_1), \ldots, E_k(M_n)$ together with an encryption of the parity-check $E_k(M_1 \oplus \ldots \oplus M_n)$ so that Bob can detect errors.
- Bob decrypts. If the parity check does not match, he sends an error message.

How can an active adversary recover the message $M$?
CPA security is not always enough!
(Some real world attacks follow a similar scenario)
Reminder: Security under Chosen Plaintext Attack (CPA)

Challenger

\[ k \xleftarrow{\$} \{0, 1\}^n \]

\[ b \xleftarrow{\$} \{0, 1\} \]

Adversary \( \mathcal{A} \)

\[ \xleftarrow{\$} x_1 \]

\[ E_k(x_1) \rightarrow \]

\[ \xleftarrow{\$} x_2 \]

\[ E_k(x_2) \rightarrow \]

\[ \ldots \]

\[ \xleftarrow{\$} (m_0, m_1) \]

\[ c^* = E_k(m_b) \rightarrow \]

Output \( b' \)
Reminder: Security under Chosen Plaintext Attack (CPA)

Challenger

\[ k \xleftarrow{R} \{0, 1\}^n \]

\[ b \xleftarrow{R} \{0, 1\} \]

\[
\begin{align*}
&\leftarrow x_1 \\
&E_k(x_1) \rightarrow \\
&\leftarrow x_2 \\
&E_k(x_2) \rightarrow \\
&\ldots
\end{align*}
\]

Adversary \( A \)

\[
\begin{align*}
&\leftarrow (m_0, m_1) \\
&c^* = E_k(m_b) \rightarrow \\
&\text{Output } b'
\end{align*}
\]

Security: For every PPT adversary \( \Pr[b = b'] \leq \frac{1}{2} + \text{neg}(n) \)
Security under Chosen Ciphertext Attack (CCA)

Challenger

\[ k \leftarrow \{0, 1\}^n \]

\[ b \leftarrow \{0, 1\} \]

\[ E_k(x_1), D_k(y_1) \rightarrow \]

\[ (x_1, y_1) \rightarrow \]

\[ x_2, y_2 \rightarrow \]

\[ \ldots \]

\[ (m_0, m_1) \rightarrow \]

\[ c^* = E_k(m_b) \rightarrow \]

Adversary \( \mathcal{A} \)

Output \( b' \)
Security under Chosen Ciphertext Attack (CCA)

**Challenger**

\[ k \xleftarrow{R} \{0, 1\}^n \]

\[ b \xleftarrow{R} \{0, 1\} \]

**Adversary \( A \)**

\[ \leftarrow x_1, y_1 \]

\[ E_k(x_1), D_k(y_1) \rightarrow \]

\[ \leftarrow x_2, y_2 \]

\[ \ldots \]

\[ \leftarrow (m_0, m_1) \]

\[ c^* = E_k(m_b) \rightarrow \]

**Output \( b' \)**

**Security:** For every PPT adversary \( \Pr[b = b'] \leq \frac{1}{2} + \text{neg}(n) \)
Security under Chosen Ciphertext Attack (CCA)

**Challenger**

\[ k \xleftarrow{\$} \{0, 1\}^n \]

\[ b \xleftarrow{\$} \{0, 1\} \]

**Adversary \( A \)**

\[ \leftarrow x_1, y_1 \]

\[ E_k(x_1), D_k(y_1) \rightarrow \]

\[ \leftarrow x_2, y_2 \]

\[ \ldots \]

\[ \leftarrow (m_0, m_1) \]

\[ c^* = E_k(m_b) \rightarrow \]

**Output \( b' \)**

**Security:** For every PPT adversary \( \Pr[b = b'] \leq \frac{1}{2} + \text{neg}(n) \)

- Decryption queries can be also asked **after** the challenge as long as \( y \neq c^* \).
CPA+MAC = CCA

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

\[
E'_k(m) = (C, T) = (E_{k_1}(m), \text{MAC}_{k_2}(C))
\]

\[
D'_{k_1, k_2}(C, T) = \begin{cases} D_{k_1}(C) & \text{if } T = \text{MAC}_{k_2}(C) \\ \bot & \text{otherwise} \end{cases}
\]

Thm. The scheme \((E', D')\) is CCA secure.

Proof idea: Assume a Chosen Ciphertext Attacker. Decryption query \(y_i\) is useful if it does not equal to an outcome of a previous encryption query. Useful queries are (almost always) answered with \(\bot\), otherwise the MAC is broken. With no useful queries, the decryption oracle isn’t really being used. We can break \(E\) via CPA.
**CPA + MAC = CCA**

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

**Construction of CCA Encryption**

\[ E'_{k_1, k_2}(M) = (C, T) \text{ where } C = E_{k_1}(M), T = \text{MAC}_{k_2}(C). \]

**Thm.** The scheme \((E', D')\) is CCA secure.

**Proof idea:** Assume a Chosen Ciphertext Attacker. Decryption query \(y_i\) is useful if it does not equal to an outcome of a previous encryption query. Useful queries are (almost always) answered with ⊥, otherwise the MAC is broken. With no useful queries, the decryption oracle isn’t really being used. We can break \(E\) via CPA.
CPA + MAC = CCA

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

### Construction of CCA Encryption

- \[ E'_{k_1, k_2}(M) = (C, T) \text{ where } C = E_{k_1}(M), \quad T = \text{MAC}_{k_2}(C). \]
- \[ D'_{k_1, k_2}(C, T) \text{ if } T = \text{MAC}_{k_2}(C) \text{ return } D_{k_1}(C), \text{ otherwise } \bot. \]
CPA + MAC = CCA

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

**Construction of CCA Encryption**

- \(E'_{k_1, k_2}(M) = (C, T)\) where \(C = E_{k_1}(M), T = \text{MAC}_{k_2}(C)\).
- \(D'_{k_1, k_2}(C, T)\) if \(T = \text{MAC}_{k_2}(C)\) return \(D_{k_1}(C)\), otherwise ⊥.
**CPA + MAC = CCA**

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

**Construction of CCA Encryption**
- \(E'_{k_1, k_2}(M) = (C, T)\) where \(C = E_{k_1}(M), T = \text{MAC}_{k_2}(C)\).
- \(D'_{k_1, k_2}(C, T)\) if \(T = \text{MAC}_{k_2}(C)\) return \(D_{k_1}(C)\), otherwise \(\perp\).

**Thm.** The scheme \((E', D')\) is CCA secure.
CPA + MAC = CCA

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

**Construction of CCA Encryption**

- \(E'_{k_1, k_2}(M) = (C, T)\) where \(C = E_{k_1}(M), T = \text{MAC}_{k_2}(C)\).
- \(D'_{k_1, k_2}(C, T)\) if \(T = \text{MAC}_{k_2}(C)\) return \(D_{k_1}(C)\), otherwise \(\bot\).

**Thm.** The scheme \((E', D')\) is CCA secure.

**Proof idea:** Assume a Chosen Ciphertext Attacker.
CPA+MAC = CCA

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

**Construction of CCA Encryption**

- \(E'_{k_1, k_2}(M) = (C, T)\) where \(C = E_{k_1}(M)\), \(T = \text{MAC}_{k_2}(C)\).
- \(D'_{k_1, k_2}(C, T)\) if \(T = \text{MAC}_{k_2}(C)\) return \(D_{k_1}(C)\), otherwise \(\perp\).

**Thm.** The scheme \((E', D')\) is CCA secure.

- **Proof idea:** Assume a Chosen Ciphertext Attacker.
- Decryption query \(y_i\) is *useful* if it does not equal to an outcome of a previous encryption query.
CPA + MAC = CCA

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

Construction of CCA Encryption

\[
E'_{k_1, k_2}(M) = (C, T) \text{ where } C = E_{k_1}(M), T = \text{MAC}_{k_2}(C).
\]

\[
D'_{k_1, k_2}(C, T) \text{ if } T = \text{MAC}_{k_2}(C) \text{ return } D_{k_1}(C), \text{ otherwise } \perp.
\]

Thm. The scheme \((E', D')\) is CCA secure.

- **Proof idea:** Assume a Chosen Ciphertext Attacker.
- Decryption query \(y_i\) is **useful** if it does not equal to an outcome of a previous encryption query.
- **Useful** queries are (almost always) answered with \(\perp\), otherwise the MAC is broken.
**CPA+MAC = CCA**

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

**Construction of CCA Encryption**

- \(E'_{k_1, k_2}(M) = (C, T)\) where \(C = E_{k_1}(M), T = \text{MAC}_{k_2}(C)\).
- \(D'_{k_1, k_2}(C, T)\) if \(T = \text{MAC}_{k_2}(C)\) return \(D_{k_1}(C)\), otherwise \(\perp\).

**Thm.** The scheme \((E', D')\) is CCA secure.

- **Proof idea:** Assume a Chosen Ciphertext Attacker.
- Decryption query \(y_i\) is useful if it does not equal to an outcome of a previous encryption query.
- **Useful** queries are (almost always) answered with \(\perp\), otherwise the MAC is broken.
- With no useful queries, the decryption oracle isn’t really being used.
**CPA + MAC = CCA**

Given CPA-secure encryption \((E, D)\) and a MAC \(\text{MAC}_k\) define \((E', D')\) as follows:

**Construction of CCA Encryption**

- \(E'_{k_1, k_2}(M) = (C, T)\) where \(C = E_{k_1}(M)\), \(T = \text{MAC}_{k_2}(C)\).
- \(D'_{k_1, k_2}(C, T)\) if \(T = \text{MAC}_{k_2}(C)\) return \(D_{k_1}(C)\), otherwise \(\perp\).

**Thm.** The scheme \((E', D')\) is CCA secure.

- **Proof idea:** Assume a Chosen Ciphertext Attacker.
- Decryption query \(y_i\) is **useful** if it does not equal to an outcome of a previous encryption query.
- **Useful** queries are (almost always) answered with \(\perp\), otherwise the MAC is broken.
- With no useful queries, the decryption oracle isn’t really being used.
- We can break \(E\) via CPA.
Summary

- Different levels of security for encryption.
Summary

- Different levels of security for encryption.
- Authentication is orthogonal to secrecy – combination is tricky.
Summary

- Different levels of security for encryption.
- Authentication is orthogonal to secrecy – combination is tricky.
- MACs and Encryption schemes can be based on PRFs/PRPs via highly efficient (practical) transformations.
Summary

- Different levels of security for encryption.
- Authentication is orthogonal to secrecy – combination is tricky.
- MACs and Encryption schemes can be based on PRFs/PRPs via highly efficient (practical) transformations.
- Good design methodology: Reduce a complicated task to a simpler task. Solve the simple task and extend the solution. (E.g., design encryption for a single-block messages and then show how to extend it to longer messages).