Cryptography

Lecture 11

Arkady Yerukhimovich

October 2, 2024

Outline

1. Lecture 10 Review
2. Secrecy vs. Integrity (Chapter 3.7)
3. Message Authentication Code (MAC) (Chapters 4.1, 4.2, 4.3.1)

Lecture 10 Review

- CCA Security
- PRF+OTP is not CCA secure
- Padding oracle attack on CBC mode

Outline

1. Lecture 10 Review
2. Secrecy vs. Integrity (Chapter 3.7)
3. Message Authentication Code (MAC) (Chapters 4.1, 4.2, 4.3.1)

Why CPA security is not enough

- We have seen two attacks against CPA-secure schemes:
  1. Attack against PRF+OTP
  2. Padding oracle attack against CBC-mode encryption

Why CPA security is not enough

- We have seen two attacks against CPA-secure schemes:
  1. Attack against PRF+OTP
  2. Padding oracle attack against CBC-mode encryption
- In both these attacks, $\mathcal{A}$ modifies received ciphertext to something whose decryption reveals information about original message

Why CPA security is not enough

- We have seen two attacks against CPA-secure schemes:
  1. Attack against PRF+OTP
  2. Padding oracle attack against CBC-mode encryption
- In both these attacks, $\mathcal{A}$ modifies received ciphertext to something whose decryption reveals information about original message
- This is called *malleability*

Why CPA security is not enough

- We have seen two attacks against CPA-secure schemes:
  1. Attack against PRF+OTP
  2. Padding oracle attack against CBC-mode encryption
- In both these attacks, $\mathcal{A}$ modifies received ciphertext to something whose decryption reveals information about original message
- This is called *malleability*
- Need to ensure only validly encrypted ciphertexts can be decrypted

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
  - Possible to change content of encrypted messages

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
    - Possible to change content of encrypted messages
    - Often want to know who sent the message

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
  - Possible to change content of encrypted messages
  - Often want to know who sent the message
- Need to guarantee *integrity* of messages

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
  - Possible to change content of encrypted messages
  - Often want to know who sent the message
- Need to guarantee *integrity* of messages

Example:

- Consider a transaction sent from a client to a bank

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
  - Possible to change content of encrypted messages
  - Often want to know who sent the message
- Need to guarantee *integrity* of messages

Example:

- Consider a transaction sent from a client to a bank
- The bank needs to be sure that:
  1. The transaction was sent by the user holding the account

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
  - Possible to change content of encrypted messages
  - Often want to know who sent the message
- Need to guarantee *integrity* of messages

Example:

- Consider a transaction sent from a client to a bank
- The bank needs to be sure that:
  1. The transaction was sent by the user holding the account
  2. The amounts in the transaction were not modified in transit

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
  - Possible to change content of encrypted messages
  - Often want to know who sent the message
- Need to guarantee *integrity* of messages

Example:

- Consider a transaction sent from a client to a bank
- The bank needs to be sure that:
  1. The transaction was sent by the user holding the account
  2. The amounts in the transaction were not modified in transit
- Integrity guarantees are needed even if transaction is public

Secrecy vs. Integrity

- Encryption hides content of message – confidentiality
- Often confidentiality is not enough
  - Possible to change content of encrypted messages
  - Often want to know who sent the message
- Need to guarantee *integrity* of messages

Example:

- Consider a transaction sent from a client to a bank
- The bank needs to be sure that:
  1. The transaction was sent by the user holding the account
  2. The amounts in the transaction were not modified in transit
- Integrity guarantees are needed even if transaction is public

We Need Integrity

Confidentiality alone is insufficient to secure information

Outline

MAC Intuition

Our Goal

Ensure that $\mathcal{A}$ cannot modify or create new (valid) messages without being detected.

MAC Intuition

Our Goal

Ensure that $\mathcal{A}$ cannot modify or create new (valid) messages without being detected.

- This should hold even if $\mathcal{A}$ sees many valid messages beforehand

MAC Intuition

Our Goal

Ensure that $\mathcal{A}$ cannot modify or create new (valid) messages without being detected.

- This should hold even if $\mathcal{A}$ sees many valid messages beforehand
- Cannot modify or recombine messages to produce a new one

MAC Intuition

Our Goal

Ensure that $\mathcal{A}$ cannot modify or create new (valid) messages without being detected.

- This should hold even if $\mathcal{A}$ sees many valid messages beforehand
- Cannot modify or recombine messages to produce a new one
- Need an "integrity tag" that can be used to check authenticity

MAC Functionality

A Message Authentication Scheme (MAC) consists of:

- $\mathsf{Gen}(1^n)$: Outputs key k with $|k| \geq n$ (usually $k \leftarrow \{0,1\}^n$)
- $\mathsf{Mac}_k(m)$: Outputs a tag $t \leftarrow \mathsf{Mac}_k(m)$
- $\mathsf{Verify}_k(m,t)$: Outputs 1 if t is a valid tag for m, and 0 otherwise

MAC Functionality

A Message Authentication Scheme (MAC) consists of:

- $\mathrm{Gen}(1^n)$: Outputs key k with $|k| \geq n$ (usually $k \leftarrow \{0,1\}^n$)
- $\mathrm{Mac}_k(m)$: Outputs a tag $t \leftarrow \mathrm{Mac}_k(m)$
- $\mathrm{Verify}_k(m,t)$: Outputs 1 if t is a valid tag for m, and 0 otherwise

Correctness

For all k output by Gen and all messages m, $\mathrm{Verify}_k(m, \mathrm{Mac}_k(m)) = 1$

MAC Functionality

A Message Authentication Scheme (MAC) consists of:

- $\mathsf{Gen}(1^n)$: Outputs key k with $|k| \geq n$ (usually $k \leftarrow \{0,1\}^n$)
- $\mathsf{Mac}_k(m)$: Outputs a tag $t \leftarrow \mathsf{Mac}_k(m)$
- $\mathsf{Verify}_k(m,t)$: Outputs 1 if t is a valid tag for m, and 0 otherwise

Correctness

For all k output by Gen and all messages m, $\mathsf{Verify}_k(m, \mathsf{Mac}_k(m)) = 1$

Canonical Verify

If Mac is deterministic, Verify can compute $\mathsf{Mac}_k(m)$, check equality to t.

MAC Unforgeability

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ be a MAC. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

MAC Unforgeability

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ be a MAC. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{MacForge}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$, and gives $\mathcal{A}$ an oracle $\mathsf{Mac}_k(\cdot)$

MAC Unforgeability

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ be a MAC. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{MacForge}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$, and gives $\mathcal{A}$ an oracle $\mathsf{Mac}_k(\cdot)$
- $\mathcal{A}^{\mathsf{Mac}_k(\cdot)}(1^n)$ outputs (m, t) (let Q be set of Mac queries made by $\mathcal{A}$)

MAC Unforgeability

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ be a MAC. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{MacForge}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$, and gives $\mathcal{A}$ an oracle $\mathsf{Mac}_k(\cdot)$
- $\mathcal{A}^{\mathsf{Mac}_k(\cdot)}(1^n)$ outputs (m, t) (let Q be set of Mac queries made by $\mathcal{A}$)
- We say that $\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if
  - $\mathsf{Verify}_k(m, t) = 1$
  - $m \notin Q$

MAC Unforgeability

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ be a MAC. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{MacForge}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$, and gives $\mathcal{A}$ an oracle $\mathsf{Mac}_k(\cdot)$
- $\mathcal{A}^{\mathsf{Mac}_k(\cdot)}(1^n)$ outputs (m, t) (let Q be set of Mac queries made by $\mathcal{A}$)
- We say that $\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if
  - $\mathsf{Verify}_k(m, t) = 1$
  - $m \notin Q$ $(m, t) \notin Q$

Definition: A MAC $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is *unforgeable* if for all PPT $\mathcal{A}$ it holds that

$$\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n)$$

Observations and Variants

- $\mathcal{A}$ can query for a mac on any message of its choice

Observations and Variants

- $\mathcal{A}$ can query for a mac on any message of its choice
- $\mathcal{A}$ WINs if he outputs valid tag on any message *he hasn't queried*

Observations and Variants

- $\mathcal{A}$ can query for a mac on any message of its choice
- $\mathcal{A}$ WINs if he outputs valid tag on any message *he hasn't queried*
- This does not prevent replay attacks, generating new tag t' for message m for which $\mathcal{A}$ has a tag t is not a forgery

Observations and Variants

- $\mathcal{A}$ can query for a mac on any message of its choice
- $\mathcal{A}$ WINs if he outputs valid tag on any message *he hasn't queried*
- This does not prevent replay attacks, generating new tag t' for message m for which $\mathcal{A}$ has a tag t is not a forgery

Strong MAC

Same definition as before, except Q contains (m, t) pairs and $\mathcal{A}$ wins if $(m, t) \notin Q$.

- Now $\mathcal{A}$ WINs if he produces a new tag even for m for which he already has a tag

Observations and Variants

- $\mathcal{A}$ can query for a mac on any message of its choice
- $\mathcal{A}$ WINs if he outputs valid tag on any message *he hasn't queried*
- This does not prevent replay attacks, generating new tag t' for message m for which $\mathcal{A}$ has a tag t is not a forgery

Strong MAC

Same definition as before, except Q contains (m, t) pairs and $\mathcal{A}$ wins if $(m, t) \notin Q$.

- Now $\mathcal{A}$ WINs if he produces a new tag even for m for which he already has a tag

- Observation: Let Π be a secure MAC that uses *canonical verify*, then Π is a strong MAC.

Unforgeable MAC from a PRF

PRF-based MAC (Fixed Length MAC)

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$

Unforgeable MAC from a PRF

PRF-based MAC (Fixed Length MAC)

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m)$: Given $m \in \{0,1\}^n$, compute $t = F_k(m)$

PRF-based MAC (Fixed Length MAC)

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m)$: Given $m \in \{0,1\}^n$, compute $t = F_k(m)$
- $\mathsf{Verify}_k(m,t)$: Compute $t' = F_k(m)$ output 1 iff $t = t'$

Unforgeable MAC from a PRF

PRF-based MAC (Fixed Length MAC)

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m)$: Given $m \in \{0,1\}^n$, compute $t = F_k(m)$
- $\mathsf{Verify}_k(m,t)$: Compute $t' = F_k(m)$ output 1 iff $t = t'$

Observations:

- Since $F_k(m)$ is unpredictable (without knowing k), $\mathcal{A}$ can't find tag for new message

Unforgeable MAC from a PRF

PRF-based MAC (Fixed Length MAC)

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m)$: Given $m \in \{0,1\}^n$, compute $t = F_k(m)$
- $\mathsf{Verify}_k(m,t)$: Compute $t' = F_k(m)$ output 1 iff $t = t'$

Observations:

- Since $F_k(m)$ is unpredictable (without knowing k), $\mathcal{A}$ can't find tag for new message
- We use canonical verify, so this is strong MAC

Unforgeable MAC from a PRF

PRF-based MAC (Fixed Length MAC)

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m)$: Given $m \in \{0,1\}^n$, compute $t = F_k(m)$
- $\mathsf{Verify}_k(m,t)$: Compute $t' = F_k(m)$ output 1 iff $t = t'$

Observations:

- Since $F_k(m)$ is unpredictable (without knowing k), $\mathcal{A}$ can't find tag for new message
- We use canonical verify, so this is strong MAC
- For proof, compare to the case where f is a random function

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):

Let $\tilde{\Pi} = (\tilde{\mathsf{Gen}}, \tilde{\mathsf{Mac}}, \tilde{\mathsf{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):

Let $\tilde{\Pi} = (\tilde{\mathsf{Gen}}, \tilde{\mathsf{Mac}}, \tilde{\mathsf{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$
   - Assume $\mathcal{A}_c$ that distinguishes between Π and $\tilde{\Pi}$ in MacForge game

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):
Let $\tilde{\Pi} = (\tilde{\text{Gen}}, \tilde{\text{Mac}}, \tilde{\text{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$
   - Assume $\mathcal{A}_c$ that distinguishes between Π and $\tilde{\Pi}$ in MacForge game
   - $\mathcal{A}_r$ receives $\mathcal{O}$ which is either f or F_k. He answers $\mathcal{A}_c$'s $\text{Mac}_k(\cdot)$ queries with $t = \mathcal{O}(m)$.

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):

Let $\tilde{\Pi} = (\tilde{\text{Gen}}, \tilde{\text{Mac}}, \tilde{\text{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$
   - Assume $\mathcal{A}_c$ that distinguishes between Π and $\tilde{\Pi}$ in MacForge game
   - $\mathcal{A}_r$ receives $\mathcal{O}$ which is either f or F_k. He answers $\mathcal{A}_c$'s $\text{Mac}_k(\cdot)$ queries with $t = \mathcal{O}(m)$.
   - If $\mathcal{O} = F_k$, this is exactly Π. if $\mathcal{O} = f$, this is exactly $\tilde{\Pi}$

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):

Let $\tilde{\Pi} = (\tilde{\text{Gen}}, \tilde{\text{Mac}}, \tilde{\text{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$
   - Assume $\mathcal{A}_c$ that distinguishes between Π and $\tilde{\Pi}$ in MacForge game
   - $\mathcal{A}_r$ receives $\mathcal{O}$ which is either f or F_k. He answers $\mathcal{A}_c$'s $\text{Mac}_k(\cdot)$ queries with $t = \mathcal{O}(m)$.
   - If $\mathcal{O} = F_k$, this is exactly Π. if $\mathcal{O} = f$, this is exactly $\tilde{\Pi}$
   - If $\mathcal{A}_c$ outputs forgery, $\mathcal{A}_r$ outputs "PRF". Succeeds with same advantage that $\mathcal{A}_c$ has

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):
Let $\tilde{\Pi} = (\tilde{\mathsf{Gen}}, \tilde{\mathsf{Mac}}, \tilde{\mathsf{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$
2. Prove that $\tilde{\Pi}$ is an unforgeable MAC

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):
Let $\tilde{\Pi} = (\tilde{\mathsf{Gen}}, \tilde{\mathsf{Mac}}, \tilde{\mathsf{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$
2. Prove that $\tilde{\Pi}$ is an unforgeable MAC
   - Note that for any $m \notin Q$, $t = f(m)$ is uniformly random

Proof Sketch

Theorem

PRF-based MAC (Π) is unforgeable for n-bit messages.

Proof (sketch):

Let $\tilde{\Pi} = (\tilde{\text{Gen}}, \tilde{\text{Mac}}, \tilde{\text{Verify}})$ be Π with random function f in place of F_k.

1. Prove by reduction that for any PPT $\mathcal{A}$, its probability of forgery changes by a negligible amount when switch from Π to $\tilde{\Pi}$
2. Prove that $\tilde{\Pi}$ is an unforgeable MAC
   - Note that for any $m \notin Q$, $t = f(m)$ is uniformly random
   - Thus,

$$\Pr[\text{MacForge}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq 2^{-n}$$

Authenticating Longer Messages

- So far, only showed how to authenticate n-bit messages

Authenticating Longer Messages

- So far, only showed how to authenticate n-bit messages
- Just like with encryption, we need to be able to authenticate arbitrary length messages

Authenticating Longer Messages

- So far, only showed how to authenticate n-bit messages
- Just like with encryption, we need to be able to authenticate arbitrary length messages
- We will explore how to do this in today's quiz.