Cryptography

Lecture 12

Arkady Yerukhimovich

October 7, 2024

Outline

1. Lecture 11 Review
2. MAC Domain Extension (Chapters 4.3.2, 4.4)
3. Authenticated Encryption (Chapter 4.5)

Lecture 11 Review

- Review of padding oracle attack
- The need for integrity
- MACs – definition and construction

Announcements

1. Exam 1 will be on Wednesday, October 16
   - It will cover material through Wednesday's lecture
   - Next Monday will be a review lecture
   - You can bring 2 sheets of 8.5×11 paper with notes

Announcements

1. Exam 1 will be on Wednesday, October 16
   - It will cover material through Wednesday's lecture
   - Next Monday will be a review lecture
   - You can bring 2 sheets of 8.5×11 paper with notes
2. Research project proposals due Wednesday, October 23
   - Team members' names
   - Brief project proposal – what topic will you cover, and what about it will you be looking at
   - No more than 1 page total

Homework 3, problem 1.C

Problem: Is $F^3_k(x) = F_k(x)||F_k(F_k(x))$ a secure PRF?

Outline

An n-bit MAC

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 $t = t'$

- This MAC can only authenticate messages of up to n bits
- How can we authenticate longer messages?

Domain Extension for MAC (Try 1)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Authenticate each block separately

$$t = t_1||t_2||\cdots||t_\ell, \text{ where } t_i = \mathsf{Mac}'_k(m_i)$$

Domain Extension for MAC (Try 1)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Authenticate each block separately

$$t = t_1||t_2||\cdots||t_\ell, \text{ where } t_i = \mathsf{Mac}'_k(m_i)$$

Problem:

- $\mathcal{A}$ can reorder blocks of m
- Given $m = m_1||m_2$, $t = t_1||t_2$, output $m' = m_2||m_1$ and $t' = t_2||t_1$

Domain Extension for MAC (Try 2)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Authenticate each block together with an index indicating order

$$t = t_1||t_2||\cdots||t_\ell, \text{ where } t_i = \mathsf{Mac}'_k(i||m_i)$$

(Make blocks a little shorter to accomodate)

Domain Extension for MAC (Try 2)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Authenticate each block together with an index indicating order

$$t = t_1||t_2||\cdots||t_\ell, \text{ where } t_i = \mathsf{Mac}'_k(i||m_i)$$

(Make blocks a little shorter to accomodate)

Problem:

- $\mathcal{A}$ can truncate message m
- Given $m = m_1||m_2$, $t = \mathsf{Mac}'_k(1||m_1)||\mathsf{Mac}'_k(2||m_2)$, output $m' = m_1$ and $t' = \mathsf{Mac}'_k(1||m_1)$

Domain Extension for MAC (Try 3)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Authenticate message length in each block

$$t = t_1||t_2||\cdots||t_\ell, \text{ where } t_i = \mathsf{Mac}'_k(\ell||i||m_i)$$

Domain Extension for MAC (Try 3)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Authenticate message length in each block

$$t = t_1||t_2||\cdots||t_\ell, \text{ where } t_i = \mathsf{Mac}'_k(\ell||i||m_i)$$

Problem:

- $\mathcal{A}$ can mix and match tags from two different messages m, m'
- Given
  $m = m_1||m_2$, $t = \mathsf{Mac}'_k(2||1||m_1)||\mathsf{Mac}'_k(2||2||m_2)$, and
  $m' = m'_1||m'_2$, $t' = \mathsf{Mac}'_k(2||1||m'_1)||\mathsf{Mac}'_k(2||2||m'_2)$
  output $\overline{m} = m_1||m'_2$ and $\overline{t} = \mathsf{Mac}'_k(2||1||m_1)||\mathsf{Mac}'_k(2||2||m'_2)$

Domain Extension for MAC (Try 4)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathrm{Gen}', \mathrm{Mac}', \mathrm{Verify}')$ be an n-bit MAC

Include random message identifier in each block:

- Parse m as $m_1||m_2||\cdots||m_\ell$ with each m_i of length $n/4$
- $r \leftarrow \{0,1\}^{n/4}$ - message id
- Compute $t_i = \mathrm{Mac}'_k(r||\ell||i||m_i)$

Domain Extension for MAC (Try 4)

Starting Point

- Let $m = m_1||m_2|| \cdots ||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Include random message identifier in each block:

- Parse m as $m_1||m_2|| \cdots ||m_\ell$ with each m_i of length $n/4$
- $r \leftarrow \{0,1\}^{n/4}$ - message id
- Compute $t_i = \mathsf{Mac}'_k(r||\ell||i||m_i)$

Theorem: This is secure arbitrary-length MAC

Domain Extension for MAC (Try 4)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Include random message identifier in each block:

- Parse m as $m_1||m_2||\cdots||m_\ell$ with each m_i of length $n/4$
- $r \leftarrow \{0,1\}^{n/4}$ - message id
- Compute $t_i = \mathsf{Mac}'_k(r||\ell||i||m_i)$

Theorem: This is secure arbitrary-length MAC
Proof (in book):

- Security of Π' means that $\mathcal{A}$ cannot make a new block with valid tag
- We've prevented the attacks we discussed
- These are (essentially) the only possible attacks

Domain Extension for MAC (Try 4)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathrm{Gen}', \mathrm{Mac}', \mathrm{Verify}')$ be an n-bit MAC

Include random message identifier in each block:

- Parse m as $m_1||m_2||\cdots||m_{4\ell}$ with each m_i of length $n/4$
- $r \leftarrow \{0,1\}^{n/4}$ - message id
- Compute $t_i = \mathrm{Mac}'_k(r||4\ell||i||m_i)$

The Problem:

Domain Extension for MAC (Try 4)

Starting Point

- Let $m = m_1||m_2||\cdots||m_\ell$, where each m_i is n bits
- Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ be an n-bit MAC

Include random message identifier in each block:

- Parse m as $m_1||m_2||\cdots||m_{4\ell}$ with each m_i of length $n/4$
- $r \leftarrow \{0,1\}^{n/4}$ - message id
- Compute $t_i = \mathsf{Mac}'_k(r||4\ell||i||m_i)$

The Problem:

This requires

- $|t| = 4\ell n$ bits
- 4ℓ calls to PRF

Question: Can we do domain extension more efficiently?

Outline

Requirements for Secure Communication

We want to build a "secure" communication channel (between A and B):

Requirements for Secure Communication

We want to build a "secure" communication channel (between A and B):

- Content of message should remain confidential, even if $\mathcal{A}$ can get decryptions of some messages

Requirements for Secure Communication

We want to build a "secure" communication channel (between A and B):

- Content of message should remain confidential, even if $\mathcal{A}$ can get decryptions of some messages
- $\mathcal{A}$ cannot modify ciphertext in transit without detection

Requirements for Secure Communication

We want to build a "secure" communication channel (between A and B):

- Content of message should remain confidential, even if $\mathcal{A}$ can get decryptions of some messages
- $\mathcal{A}$ cannot modify ciphertext in transit without detection
- $\mathcal{A}$ cannot produce new valid ciphertext on behalf of either party

Requirements for Secure Communication

We want to build a "secure" communication channel (between A and B):

- Content of message should remain confidential, even if $\mathcal{A}$ can get decryptions of some messages
- $\mathcal{A}$ cannot modify ciphertext in transit without detection
- $\mathcal{A}$ cannot produce new valid ciphertext on behalf of either party

Question

Does CCA-secure encryption achieve these properties?

Requirements for Secure Communication

We want to build a "secure" communication channel (between A and B):

- Content of message should remain confidential, even if $\mathcal{A}$ can get decryptions of some messages
- $\mathcal{A}$ cannot modify ciphertext in transit without detection
- $\mathcal{A}$ cannot produce new valid ciphertext on behalf of either party

Question

Does CCA-secure encryption achieve these properties?

- CCA-security achieves item 1
- But, it does not prevent $\mathcal{A}$ from producing valid ciphertexts unrelated to the challenge
- This violates bullet 3

Unforgeable Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

Unforgeable Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

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

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

Unforgeable Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

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

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

Unforgeable Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

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

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

Unforgeable Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

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

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

Unforgeable Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

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

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

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

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

Authenticated Encryption

Definition: Π is an *authenticated encryption scheme* if it is:

- CCA-secure
- Unforgeable

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\text{Gen}, \text{Mac}, \text{Verify})$ is a secure MAC
- $\Pi_E = (\text{Gen}, \text{Enc}, \text{Dec})$ is a CPA-secure encryption scheme

Try 1: Encrypt and Authenticate

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 1: Encrypt and Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 1: Encrypt and Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 1: Encrypt and Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$
- $t \leftarrow \mathsf{Mac}_{k_M}(m)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 1: Encrypt and Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$
- $t \leftarrow \mathsf{Mac}_{k_M}(m)$
- Output (c, t)

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 1: Encrypt and Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$
- $t \leftarrow \mathsf{Mac}_{k_M}(m)$
- Output (c, t)

A problem:

- t may not provide confidentiality of m. In particular, if Mac is deterministic $t = \mathsf{Mac}_{k_M}(m)$ may leak info about m

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 2: Authenticate then Encrypt

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 2: Authenticate then Encrypt

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 2: Authenticate then Encrypt

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $t \leftarrow \mathsf{Mac}_{k_M}(m)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 2: Authenticate then Encrypt

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $t \leftarrow \mathsf{Mac}_{k_M}(m)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m||t)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 2: Authenticate then Encrypt

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $t \leftarrow \mathsf{Mac}_{k_M}(m)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m||t)$
- Output (c)

Pro: t no longer revealed in the clear

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 2: Authenticate then Encrypt

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $t \leftarrow \mathsf{Mac}_{k_M}(m)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m||t)$
- Output (c)

Pro: t no longer revealed in the clear

A problem:

- May allow padding oracle attack. Especially, if provide decryption error messages ("bad padding" vs. "MAC failed")

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 3: Encrypt then Authenticate

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 3: Encrypt then Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 3: Encrypt then Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 3: Encrypt then Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$
- $t \leftarrow \mathsf{Mac}_{k_M}(c)$

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 3: Encrypt then Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$
- $t \leftarrow \mathsf{Mac}_{k_M}(c)$
- Output (c, t)

Constructing Authenticated Encryption

Building blocks:

- $\Pi_M = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is a secure MAC
- $\Pi_E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme

Try 3: Encrypt then Authenticate

- $k_E \leftarrow \mathsf{Gen}(1^n)$, $k_M \leftarrow \mathsf{Gen}(1^n)$
- $c \leftarrow \mathsf{Enc}_{k_E}(m)$
- $t \leftarrow \mathsf{Mac}_{k_M}(c)$
- Output (c, t)

Use encrypt-then-authenticate

Encrypt then authenticate is best way to construct authenticated encryption

Proof Sketch for Encrypt then Authenticate

Assumptions and Notation:

- Assume Mac is a strong MAC and Enc is a CPA-secure encryption scheme
- Say that $C = (c, t)$ is *valid* if t is a valid tag on c

Proof Sketch for Encrypt then Authenticate

Assumptions and Notation:

- Assume Mac is a strong MAC and Enc is a CPA-secure encryption scheme
- Say that $C = (c, t)$ is *valid* if t is a valid tag on c

Proof:

1. By strong security of MAC, PPT $\mathcal{A}_c$ cannot generate any valid $C' = (c', t')$ that it did not receive from $\mathsf{Enc}(\cdot)$ oracle query

Proof Sketch for Encrypt then Authenticate

Assumptions and Notation:

- Assume Mac is a strong MAC and Enc is a CPA-secure encryption scheme
- Say that $C = (c, t)$ is *valid* if t is a valid tag on c

Proof:

1. By strong security of MAC, PPT $\mathcal{A}_c$ cannot generate any valid $C' = (c', t')$ that it did not receive from $\mathrm{Enc}(\cdot)$ oracle query
2. Decryption oracle is useless (so, $\mathcal{A}_r$ can answer Dec queries)

Proof Sketch for Encrypt then Authenticate

Assumptions and Notation:

- Assume Mac is a strong MAC and Enc is a CPA-secure encryption scheme
- Say that $C = (c, t)$ is *valid* if t is a valid tag on c

Proof:

1. By strong security of MAC, PPT $\mathcal{A}_c$ cannot generate any valid $C' = (c', t')$ that it did not receive from $\mathsf{Enc}(\cdot)$ oracle query
2. Decryption oracle is useless (so, $\mathcal{A}_r$ can answer Dec queries)
    - For every query $\mathsf{Dec}(c', t')$ made by $\mathcal{A}_c$, $\mathcal{A}_r$ knows that either:
        - (c', t') was output by prior Enc query, or
        - $\mathsf{Dec}(c', t') = \perp$

Proof Sketch for Encrypt then Authenticate

Assumptions and Notation:

- Assume Mac is a strong MAC and Enc is a CPA-secure encryption scheme
- Say that $C = (c, t)$ is *valid* if t is a valid tag on c

Proof:

1. By strong security of MAC, PPT $\mathcal{A}_c$ cannot generate any valid $C' = (c', t')$ that it did not receive from $\text{Enc}(\cdot)$ oracle query
2. Decryption oracle is useless (so, $\mathcal{A}_r$ can answer Dec queries)
    - For every query $\text{Dec}(c', t')$ made by $\mathcal{A}_c$, $\mathcal{A}_r$ knows that either:
        - (c', t') was output by prior Enc query, or
        - $\text{Dec}(c', t') = \perp$
    - $\mathcal{A}_r$ answers $\mathcal{A}_c$'s Enc queries using own oracle, storing queries and answers

Proof Sketch for Encrypt then Authenticate

Assumptions and Notation:

- Assume Mac is a strong MAC and Enc is a CPA-secure encryption scheme
- Say that $C = (c, t)$ is *valid* if t is a valid tag on c

Proof:

1. By strong security of MAC, PPT $\mathcal{A}_c$ cannot generate any valid $C' = (c', t')$ that it did not receive from $\mathrm{Enc}(\cdot)$ oracle query
2. Decryption oracle is useless (so, $\mathcal{A}_r$ can answer Dec queries)
   - For every query $\mathrm{Dec}(c', t')$ made by $\mathcal{A}_c$, $\mathcal{A}_r$ knows that either:
     - (c', t') was output by prior Enc query, or
     - $\mathrm{Dec}(c', t') = \perp$
   - $\mathcal{A}_r$ answers $\mathcal{A}_c$'s Enc queries using own oracle, storing queries and answers
   - on Dec query from $\mathcal{A}_c$, see if it is result of prior Enc query, if so output m, if not output $\perp$

Proof Sketch for Encrypt then Authenticate

Assumptions and Notation:

- Assume Mac is a strong MAC and Enc is a CPA-secure encryption scheme
- Say that $C = (c, t)$ is *valid* if t is a valid tag on c

Proof:

1. By strong security of MAC, PPT $\mathcal{A}_c$ cannot generate any valid $C' = (c', t')$ that it did not receive from $\text{Enc}(\cdot)$ oracle query
2. Decryption oracle is useless (so, $\mathcal{A}_r$ can answer Dec queries)
   - For every query $\text{Dec}(c', t')$ made by $\mathcal{A}_c$, $\mathcal{A}_r$ knows that either:
     - (c', t') was output by prior Enc query, or
     - $\text{Dec}(c', t') = \perp$
   - $\mathcal{A}_r$ answers $\mathcal{A}_c$'s Enc queries using own oracle, storing queries and answers
   - on Dec query from $\mathcal{A}_c$, see if it is result of prior Enc query, if so output m, if not output $\perp$
   - $\mathcal{A}_r$ does not need a Dec oracle, so CPA-security implies CCA-security

MAC and Enc Must Use Independent Keys

- It is tempting to use the same key k for both Enc and Mac.

MAC and Enc Must Use Independent Keys

- It is tempting to use the same key k for both Enc and Mac.
- DO NOT DO THIS!

MAC and Enc Must Use Independent Keys

- It is tempting to use the same key k for both Enc and Mac.
- DO NOT DO THIS!

A bad example:

- Let $\mathsf{Enc}_k(m) = F_k(r||m)$

MAC and Enc Must Use Independent Keys

- It is tempting to use the same key k for both Enc and Mac.
- DO NOT DO THIS!

A bad example:

- Let $\mathrm{Enc}_k(m) = F_k(r||m)$
- Let $\mathrm{Mac}_k(c) = F_k^{-1}(c)$ (secure MAC if F_k is strong PRP)

MAC and Enc Must Use Independent Keys

- It is tempting to use the same key k for both Enc and Mac.
- DO NOT DO THIS!

A bad example:

- Let $\mathsf{Enc}_k(m) = F_k(r||m)$
- Let $\mathsf{Mac}_k(c) = F_k^{-1}(c)$ (secure MAC if F_k is strong PRP)
- Output is $(\mathsf{Enc}_k(m), \mathsf{Mac}_k(\mathsf{Enc}_k(m)))$

MAC and Enc Must Use Independent Keys

- It is tempting to use the same key k for both Enc and Mac.
- DO NOT DO THIS!

A bad example:

- Let $\mathrm{Enc}_k(m) = F_k(r||m)$
- Let $\mathrm{Mac}_k(c) = F_k^{-1}(c)$ (secure MAC if F_k is strong PRP)
- Output is $(\mathrm{Enc}_k(m), \mathrm{Mac}_k(\mathrm{Enc}_k(m)))$

The Problem:

$$\begin{aligned}(\mathrm{Enc}_k(m), \mathrm{Mac}_k(\mathrm{Enc}_k(m))) &= F_k(r||m), F_k^{-1}(F_k(r||m)) \\ &= F_k(r||m), r||m\end{aligned}$$

MAC and Enc Must Use Independent Keys

- It is tempting to use the same key k for both Enc and Mac.
- DO NOT DO THIS!

A bad example:

- Let $\mathrm{Enc}_k(m) = F_k(r||m)$
- Let $\mathrm{Mac}_k(c) = F_k^{-1}(c)$ (secure MAC if F_k is strong PRP)
- Output is $(\mathrm{Enc}_k(m), \mathrm{Mac}_k(\mathrm{Enc}_k(m)))$

The Problem:

$$\begin{aligned}(\mathrm{Enc}_k(m), \mathrm{Mac}_k(\mathrm{Enc}_k(m))) &= F_k(r||m), F_k^{-1}(F_k(r||m)) \\ &= F_k(r||m), r||m\end{aligned}$$

Insecure

Using the same key reveals the message

The Status of Authenticated Encryption

- Authenticated encryption has become standard for secure communication
- Special modes of operations for authenticated encryption of arbitrary length messages:
  - Galois Counter Mode (GCM) - standardized by NIST

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Basic Idea

Use authenticated encryption to encrypt every message between A and B.

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Basic Idea

Use authenticated encryption to encrypt every message between A and B.

The Problems: To ensure both parties receive only correct content, need to deal with following attacks

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Basic Idea

Use authenticated encryption to encrypt every message between A and B.

The Problems: To ensure both parties receive only correct content, need to deal with following attacks

- Re-ordering attack – $\mathcal{A}$ swaps order of messages over the network

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Basic Idea

Use authenticated encryption to encrypt every message between A and B.

The Problems: To ensure both parties receive only correct content, need to deal with following attacks

- Re-ordering attack – $\mathcal{A}$ swaps order of messages over the network
- Replay attack – $\mathcal{A}$ resends valid ct sent before by one of the parties

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Basic Idea

Use authenticated encryption to encrypt every message between A and B.

The Problems: To ensure both parties receive only correct content, need to deal with following attacks

- Re-ordering attack – $\mathcal{A}$ swaps order of messages over the network
- Replay attack – $\mathcal{A}$ resends valid ct sent before by one of the parties
- Reflection attack - $\mathcal{A}$ sends message from A to B back to A.

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Basic Idea

Use authenticated encryption to encrypt every message between A and B.

The Problems: To ensure both parties receive only correct content, need to deal with following attacks

- Re-ordering attack – $\mathcal{A}$ swaps order of messages over the network
- Replay attack – $\mathcal{A}$ resends valid ct sent before by one of the parties
- Reflection attack - $\mathcal{A}$ sends message from A to B back to A.

Solution: Use a session counter and directionality bit

- $c \leftarrow \mathsf{Enc}_k(b_{A,B}||ctr_{A,B}||m)$

Building a Secure Communication Session

Suppose, A and B (sharing a key k) want to establish a secure communication session:

- Send many messages back and forth
- Prevent $\mathcal{A}$ from interfering

Basic Idea

Use authenticated encryption to encrypt every message between A and B.

The Problems: To ensure both parties receive only correct content, need to deal with following attacks

- Re-ordering attack – $\mathcal{A}$ swaps order of messages over the network
- Replay attack – $\mathcal{A}$ resends valid ct sent before by one of the parties
- Reflection attack - $\mathcal{A}$ sends message from A to B back to A.

Solution: Use a session counter and directionality bit

- $c \leftarrow \mathrm{Enc}_k(b_{A,B}||ctr_{A,B}||m)$
- Requires A and B to be stateful