

CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2025 Lecture 5

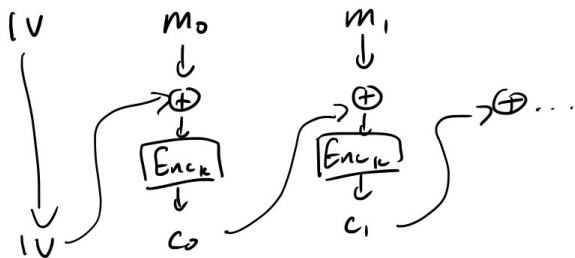
Announcements

1. HW 2 is due next lecture!
2. HW 3 is online, due before class in 1.5 weeks.

Last time: Chosen plaintext attacks

This time: Fun with CBC mode, malleability and message integrity

Cipher block chaining (CBC) mode



1. IV has same length as block length.
2. $c_i = Enc_k(c_{i-1} \oplus m_i)$
3. Output $(IV, c_0, c_1, c_2, \dots)$.

IV should be random.

CBC mode is CPA-secure, but suffers from implementation vulnerabilities.

Message padding for encryption

Q: How do you encrypt odd-length messages with a fixed-length block cipher?

A: Pad message somehow.

Message padding for encryption

Q: How do you encrypt odd-length messages with a fixed-length block cipher?

A: Pad message somehow.

Example: PKCS 7 padding.

- If message is b bytes short of 128-bit block, append b bytes $bbb \dots b$ to make block multiple of 128.
- Decryptor checks and strips off padding.

Message padding for encryption

Q: How do you encrypt odd-length messages with a fixed-length block cipher?

A: Pad message somehow.

Example: PKCS 7 padding.

- If message is b bytes short of 128-bit block, append b bytes $bbb \dots b$ to make block multiple of 128.
- Decryptor checks and strips off padding.

Q: What do you do if padding is incorrect?

A: Throw an error?

Padding oracle attacks

Vaudenay 2002: CBC mode encryption is insecure if attacker can distinguish bad from good message padding

Padding Oracle Attack

Want to decrypt $(IV, c_1, \dots, c_n) = \text{Enc}_k(m_1, \dots, m_n)$.

Have oracle that given $(IV', c'_1, \dots, c'_\ell)$ will:

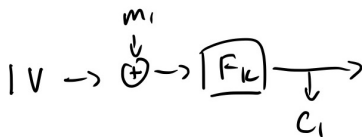
1. Compute $(m'_1, \dots, m'_\ell) = \text{Dec}_k(IV', c'_1, \dots, c'_\ell)$
2. Return

$$\begin{cases} \text{valid} & \text{if } m'_\ell \text{ ends in valid padding} \\ \text{invalid} & \text{if } m'_\ell \text{ doesn't end in valid padding} \end{cases}$$

Vaudenay CBC padding oracle attack

Input: Ciphertext $(IV, c_1, c_2, \dots, c_n)$

1. Attacker sends $(IV \oplus 00 \dots 00t, c_1) = F_k(IV \oplus m_1 \oplus 00 \dots 00t)$ to decryption padding oracle.

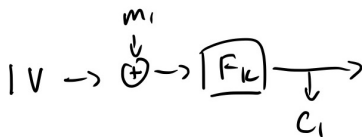


oracle returns $\left\{ \begin{array}{l} \text{valid if } m_1 \oplus 000 \dots 0t = \dots 1 \text{ (most likely)} \\ \dots 22 \\ \dots 333 \\ \vdots \\ \text{invalid otherwise} \end{array} \right.$

Vaudenay CBC padding oracle attack

Input: Ciphertext $(IV, c_1, c_2, \dots, c_n)$

1. Attacker sends $(IV \oplus 00 \dots 00t, c_1) = F_k(IV \oplus m_1 \oplus 00 \dots 00t)$ to decryption padding oracle.



oracle returns $\left\{ \begin{array}{l} \text{valid if } m_1 \oplus 000 \dots 0t = \dots 1 \text{ (most likely)} \\ \dots 22 \\ \dots 333 \\ \vdots \\ \text{invalid otherwise} \end{array} \right.$

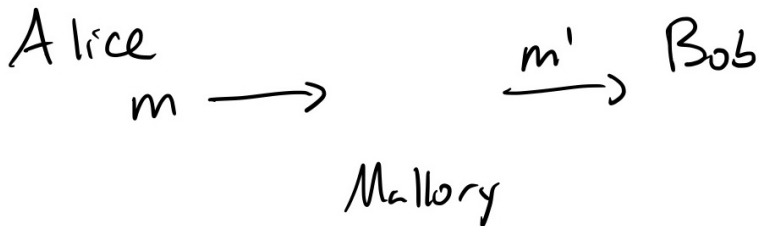
2. Try all 256 values of t .

$$IV[0] \oplus m_1[0] \oplus t = 1$$

↑ known ↑ unknown ↑ known ↑ known

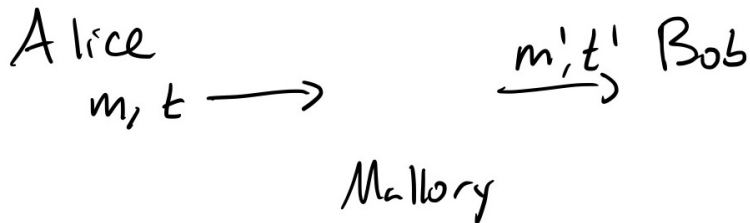
3. Good probability of learning value
4. Iterate for successive bytes of ciphertext.
5. $256n$ query complexity to learn n bytes of plaintext.

Message Malleability and Integrity



Independent of whether messages remain confidential, we want to ensure that they can't be modified by a third party in transit.

Message Authentication Codes



Solution: Add a special tag to ensure integrity of the message.

Cryptographic and non-cryptographic integrity checking

Non-cryptographic protocols also often include integrity checks:

- Ethernet uses CRC32, a 32-bit checksum
- TCP has a 16-bit checksum

These algorithms protect against *random* errors, but they are public and do not include keys so can be forged by a malicious adversary.

Cryptographic and non-cryptographic integrity checking

In order to protect against malicious forgery, cryptographic integrity checks must use a secret key.

Encryption does not suffice for integrity checking.

Cryptographic and non-cryptographic integrity checking

In order to protect against malicious forgery, cryptographic integrity checks must use a secret key.

Encryption does not suffice for integrity checking.

Example: Stream cipher encryption.

$$\text{Enc}_k(m) = G(k) \oplus m = c$$

$$\text{Dec}_k(c \oplus \text{anything}) = m \oplus \text{anything}$$

Message Authentication Codes

Definition

- Key generation algorithm generates k
- Tag generation: $\text{Mac}_k(m) \rightarrow t$
- Tag verification:

$$\text{Verify}_k(m, t) = \begin{cases} \text{accept} & \text{if tag is valid} \\ \text{reject} & \text{if tag is invalid} \end{cases}$$

- Correctness: $\Pr[\text{Verify}_k(m, \text{Mac}_k(m)) = \text{accept}] = 1$

MAC Security: Existential MAC forgery

A
oracle access to $\text{Mac}_k(\cdot)$



C
generate random k

A wins if

- $\text{Verify}_k(m, t) = 1$
- A never queried m

Definition

A MAC construction (Mac, Verify) is existentially unforgeable under a chosen-message attack if the probability that A wins is negligible.

"Strongly secure" MACs

A
oracle access to $\text{Mac}_k(\cdot)$



$(m, t) \rightarrow$

C
generate random k

A wins if

- $\text{Verify}_k(m, t) = 1$
- (m, t) not among signed pairs

This is equivalent to the previous definition for deterministic MACs. The adversary can query the target message but not return one of the responses.

Constructing a MAC from a PRF

Let F be a PRF. Then we can construct a MAC as follows:

- Key generation: $k \in_R \{0, 1\}^n$
- $\text{Mac}_k(m) = F_k(m)$
-

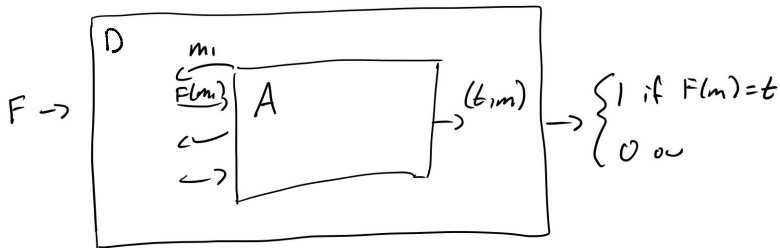
$$\text{Verify}_k(m, t) = \begin{cases} \text{accept} & \text{if } F_k(m) = t \\ \text{reject} & \text{otherwise} \end{cases}$$

Theorem

The PRF MAC construction is secure.

Proof.

Assume construction not secure, construct PRF distinguisher.



1. If F is a truly random function, then $\Pr[A \text{ succeeds}] = 2^{-n} = \Pr[D(f) = 1]$
2. If F PRF, $\Pr[D(F_k) = 1] = d > \text{negligible}$ by assumption.

$$|\Pr[D(F_k) = 1] - \Pr[D(f) = 1]| = d - 2^{-n} > \text{negligible}$$

CBC-MAC

Can use CBC construction to construct a fixed-length MAC for arbitrary length messages.

- $k \in_R \{0, 1\}^n$
- Input $m = m_1 \dots m_\ell$. To compute $\text{Mac}_k(m)$:
 1. $t_0 = 0^n$ (fixed, non-random value)
 2. For $i = 1, \dots, \ell$ $t_i = F_k(t_{i-1} \oplus m_i)$
 3. Output last block t_ℓ

-

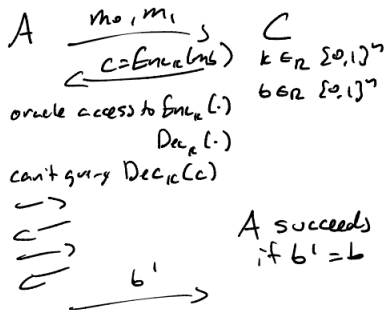
$$\text{Verify}_k(m, t) = \begin{cases} \text{accept} & \text{if } \text{Mac}_k(m) = t \\ \text{reject} & \text{otherwise} \end{cases}$$

Theorem

If F is a secure PRF, then CBC-MAC is a secure fixed-length MAC for arbitrary-length messages

Chosen Ciphertext Attacks

CCA Indistinguishability Experiment

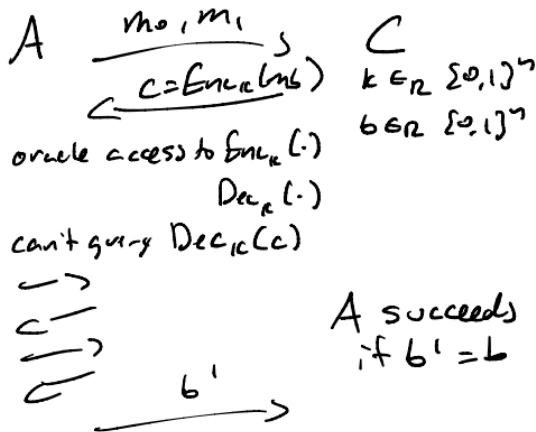


Definition ("CCA-secure")

(Enc, Dec) has indistinguishable encryptions under a chosen ciphertext attack if \forall efficient adversaries A , $\Pr[A \text{ succeeds}] \leq 1/2 + \epsilon \in \text{negligible}$.

Chosen Ciphertext Attacks

CCA Indistinguishability Experiment



IND-CCA1: “non-adaptive”
Decryption oracle only queried
prior to challenge ciphertext.

IND-CCA2: “adaptive” Adversary
may make further calls to
decryption oracle

How do you achieve CCA security?

Right answer in practice: Use a pre-defined authenticated encryption mode.

How do you achieve CCA security?

Right answer in practice: Use a pre-defined authenticated encryption mode.

How do you combine encryption and MAC to achieve authenticated encryption?

- Encrypt-and-MAC? Used in SSH.

$$c = \text{Enc}_{k_e}(m) \quad t = \text{Mac}_{k_m}(m) \quad \text{send } (c, t)$$

- MAC-then-encrypt? Used in SSL/TLS 1.2 and below.

$$t = \text{Mac}_{k_m}(m) \quad c = \text{Enc}_{k_e}(m||t) \quad \text{send } c$$

- Encrypt-then-MAC? Used in IPsec.

$$c = \text{Enc}_{k_e}(m) \quad t = \text{Mac}_{k_m}(c) \quad \text{send } (c, t)$$

How do you achieve CCA security?

Right answer in practice: Use a pre-defined authenticated encryption mode.

How do you combine encryption and MAC to achieve authenticated encryption?

- Encrypt-and-MAC? Used in SSH.

$$c = \text{Enc}_{k_e}(m) \quad t = \text{Mac}_{k_m}(m) \quad \text{send } (c, t)$$

- MAC-then-encrypt? Used in SSL/TLS 1.2 and below.

$$t = \text{Mac}_{k_m}(m) \quad c = \text{Enc}_{k_e}(m||t) \quad \text{send } c$$

- Encrypt-then-MAC? Used in IPsec. ✓

$$c = \text{Enc}_{k_e}(m) \quad t = \text{Mac}_{k_m}(c) \quad \text{send } (c, t)$$

Intuition: Protect integrity of ciphertext to prevent mauling.

Reminder, HW 2 is due before next lecture so we can talk about it in lecture!