CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2023 Lecture 5

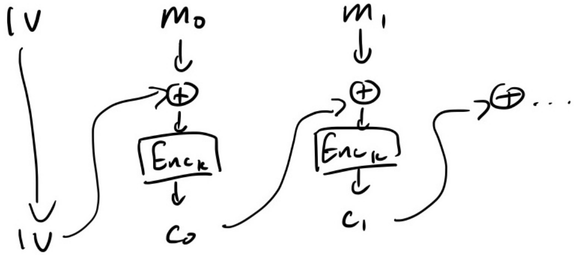
Announcements

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

Last time: Chosen plaintext attacks

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

Last time: Cipher block chaining (CBC) mode



1. IV has same length as block length.
2. $c_i = \text{Enc}_k(c_{i-1} \oplus m_i)$
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 $bbb \ldots b$ 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 $(IV, c_1, \ldots, c_n) = \mathrm{Enc}_k(m_1, \ldots, m_n)$.

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

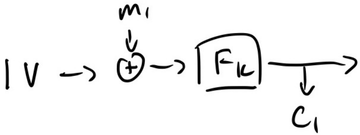
1. Compute $(m'_1, \ldots, m'_\ell) = \mathrm{Dec}_k(IV, c'_1, \ldots, c'_\ell)$
2. Return

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

Vaudenay CBC padding oracle attack

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

1. Attacker sends $(IV \oplus 00\ldots00t, c_1) = F_k(IV \oplus m_1 \oplus 00\ldots00t)$ to decryption padding oracle.

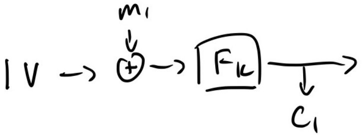


$$\text{oracle returns} \begin{cases} \texttt{valid} & \text{if } m_1 \oplus 000\ldots0t = \quad \ldots 1 \text{(most likely)} \\ & \qquad\qquad\qquad\qquad\quad\;\; \ldots 22 \\ & \qquad\qquad\qquad\qquad\quad\;\; \ldots 333 \\ & \qquad\qquad\qquad\qquad\quad\;\; \vdots \\ \texttt{invalid} & \text{otherwise} \end{cases}$$

Vaudenay CBC padding oracle attack

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

1. Attacker sends $(IV \oplus 00\ldots00t, c_1) = F_k(IV \oplus m_1 \oplus 00\ldots00t)$ to decryption padding oracle.



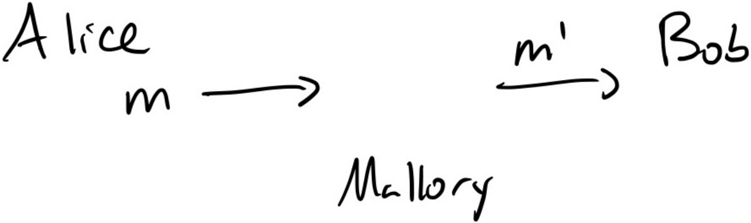
$$\text{oracle returns} \begin{cases} \texttt{valid} & \text{if } m_1 \oplus 000\ldots0t = \quad \ldots 1 (\text{most likely}) \\ & \qquad\qquad\qquad\qquad\quad\ \ldots 22 \\ & \qquad\qquad\qquad\qquad\quad\ \ldots 333 \\ & \qquad\qquad\qquad\qquad\quad\ \vdots \\ \texttt{invalid} & \text{otherwise} \end{cases}$$

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

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

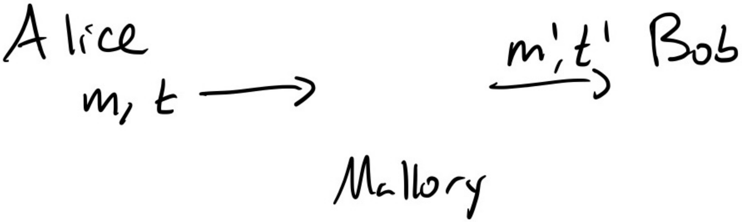
(known, unknown, known, known)

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 cheks:

- 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.

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

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

Message Authentication Codes

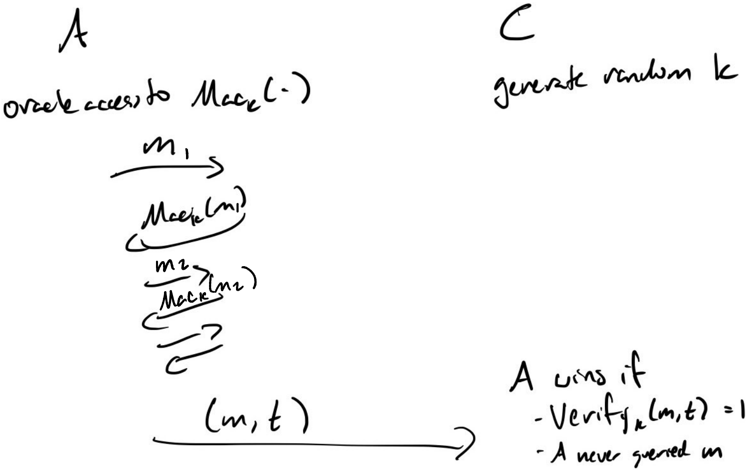
Definition

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

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

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

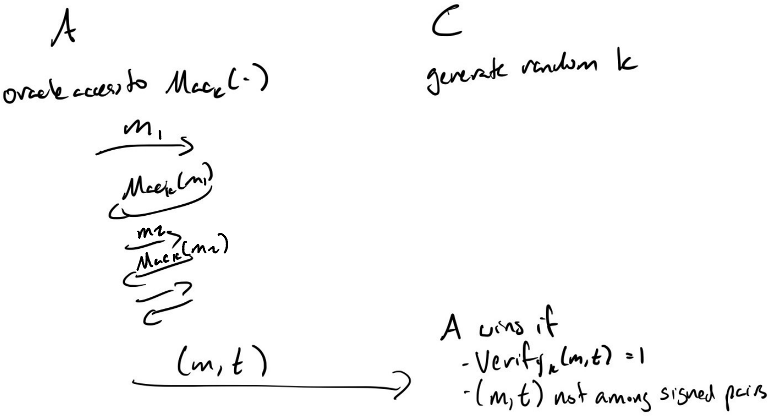
MAC Security: Existential MAC forgery



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



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:

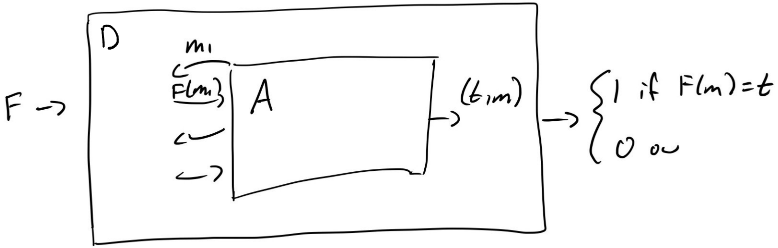
- Key generation: $k \in_R \{0,1\}^n$
- $\text{Mac}_k(m) = F_k(m)$
- $$\text{Verify}_k(m,t) = \begin{cases} \texttt{accept} & \text{if } F_k(m) = t \\ \texttt{reject} & \text{otherwise} \end{cases}$$

Theorem

The PRF MAC construction is secure.

Proof.

Assume construction not secure, construct PRF distinguisher.



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

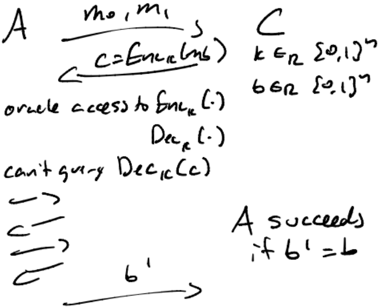
- $k \in_R \{0,1\}^n$
- Input $m = m_1 \ldots m_\ell$. To compute $\mathrm{Mac}_k(m)$:
  1. $t_0 = 0^n$ (fixed, non-random value)
  2. For $i = 1, \ldots, \ell$ $t_i = F_k(t_{i-1} \oplus m_i)$
  3. Output last block t_ℓ
- $$\mathrm{Verify}_k(m,t) = \begin{cases} \texttt{accept} & \text{if } \mathrm{Mac}_k(m) = t \\ \texttt{reject} & \text{otherwise} \end{cases}$$

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

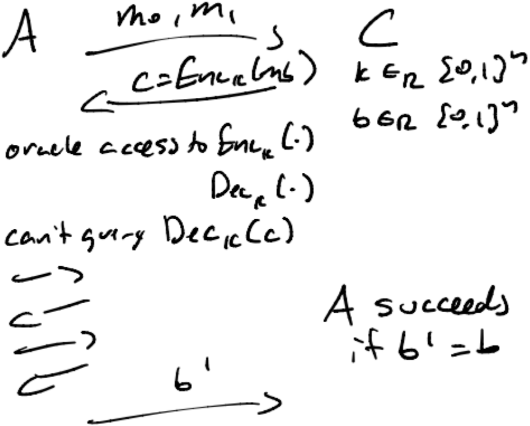


Definition ("CCA-secure")

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

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

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

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

- Encrypt-then-MAC? Used in IPsec.

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

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.

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

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

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

- Encrypt-then-MAC? Used in IPsec. ✔

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

Intuition: Protect integrity of ciphertext to prevent mauling.

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