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 \((IV, c_0, c_1, c_2, \ldots)\).

IV should be random.

CBC mode is CPA-secure, but suffers from implementation vulnerabilities.
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 $\ldots b$ 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?
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.
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.

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 \((IV, c_1, \ldots, c_n) = Enc_k(m_1, \ldots, m_n)\).

Have oracle that given \((IV', c'_\ell, \ldots, c'_\ell)\) will:
1. Compute \((m'_1, \ldots, m'_\ell) = Dec_k(IV, c'_1, \ldots, c'_\ell)\)
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 \((\text{IV}, c_1, c_2, \ldots, c_n)\)

1. Attacker sends \((\text{IV} \oplus 00 \ldots 00 t, c_1) = F_k(\text{IV} \oplus m_1 \oplus 00 \ldots 00 t)\) to decryption padding oracle.

\[
\text{IV} \rightarrow \oplus \rightarrow \boxed{F_k} \rightarrow c_1
\]

**oracle returns**
\[
\begin{cases} 
\text{valid if } m_1 \oplus 000 \ldots 0t = \ldots 1 \text{(most likely)}
\quad \ldots 22 \\
\quad \ldots 333 \\
\text{invalid otherwise}
\end{cases}
\]
Vaudenay CBC padding oracle attack

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

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

   \[
   \begin{align*}
   IV & \rightarrow \oplus \rightarrow F_k \rightarrow c_1 \\
   & \begin{cases}
   \text{valid if } m_1 \oplus 000 \ldots 0t = \ldots 1 \text{(most likely)} \\
   \ldots 22 \\
   \ldots 333 \\
   \vdots
   
   \end{cases}
   \end{align*}
   \]

   oracle returns

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.
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.
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.

\[
Enc_k(m) = G(k) \oplus m = c
\]

\[
Dec_k(c \oplus \text{anything}) = m \oplus \text{anything}
\]
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

**Definition**
A MAC construction \((\text{Mac}, \text{Verify})\) 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: $k \in_R \{0, 1\}^n$
- $\text{Mac}_k(m) = F_k(m)$
- $\text{Verify}_k(m, t) = \begin{cases} 
\text{accept} & \text{if } F_k(m) = t \\
\text{reject} & \text{otherwise}
\end{cases}$
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 \( m = m_1 \ldots m_\ell \). To compute \( \text{Mac}_k(m) \):
  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 \( t_\ell \)

•

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

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

\[ A \] \[ m_0, m_1 \] \[ \text{oracle access to Enc} \] \[ C = \text{Enc}_k(m_b) \] \[ C = \text{Dec}_k(C) \] \[ \text{can't query Dec}_k(C) \] \[ A \text{ succeeds} \]

Definition ("CCA-secure")

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

\[
A \xrightarrow{m_0, m_1} \underline{c = \text{Enc}_k(m_i)} \xrightarrow{\text{oracle access to Enc}_k(\cdot)} \text{Dec}_k(\cdot)
\]

\[
\text{can't query Dec}_k(c) \rightarrow \rightarrow c \rightarrow 61 \rightarrow A \text{ succeeds if } b' = 1
\]
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.

\[ t = \text{Mac}_{k_m}(m) \quad c = \text{Enc}_{k_e}(m \mid t) \quad \text{send} \ c \]

- 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!