Limitations of CPA security

How?
Message Authentication

Goal: If Eve changed $m$, Bob should reject
Message Authentication Codes

Syntax:
- Key space $K$
- Message space $M$
- Tag space $T$
- $MAC(k,m) \rightarrow \sigma$
- $Ver(k,m,\sigma) \rightarrow 0/1$

Correctness:
- $\forall m,k, \ Ver(k,m, \ MAC(k,m)) = 1$
\( q \)-Time MACs

\[
q \text{ times} \quad m_i \in M \quad \sigma \quad (m^*, \sigma^*)
\]

Output 1 iff:
- \( m^* \notin \{m_1, \ldots, m_q\} \)
- \( \text{Ver}(k, m^*, \sigma^*) = 1 \)

\[
q\text{CMA-Adv} = \Pr[\text{outputs } 1]
\]
Computational Security

**Definition:** \((\text{MAC}, \text{Ver})\) is \((t, q, \varepsilon)\)-secure under a chosen message attack (**CMA-secure**) if, for all running in time at most \(t\) and making at most \(q\) queries,

\[
\text{CMA-Adv}(\widehat{\mathcal{A}}) \leq \varepsilon
\]
Constructing MACs

Use a PRF

\( F:K \times M \rightarrow T \)

\( MAC(k,m) = F(k,m) \)

\( Ver(k,m,\sigma) = (F(k,m) = \sigma) \)
Theorem: If $F$ is $(t,q,\varepsilon)$-secure then $(\text{MAC},\text{Ver})$ is $(t-t',q,\varepsilon+1/|T|)$-CMA secure
CBC-MAC

Theorem: CBC-MAC is a secure PRF for fixed-length messages
Today

Other Considerations

Authenticated Encryption – combining encryption with MACs
Timing Attacks on MACs

How do you implement check $F(k,m) = \sigma$?

String comparison often optimized for performance

**Compare(A,B):**
- For $i = 1,...,A.length$
  - If $A[i] \neq B[i]$, abort and return False;
- Return True;

Time depends on number of initial bytes that match
Timing Attacks on MACs

To forge a message $m$:

For each candidate first byte $\sigma_0$:
- Query server on $(m, \sigma)$ where first byte of $\sigma$ is $\sigma_0$
- See how long it takes to reject

First byte is $\sigma_0$ that causes the longest response
- If wrong, server rejects when comparing first byte
- If right, server rejects when comparing second
Timing Attacks on MACs

To forge a message \( m \):

Now we have first byte \( \sigma_0 \)

For each candidate second byte \( \sigma_1 \):
- Query server on \( (m, \sigma) \) where first two bytes of \( \sigma \) are \( \sigma_0, \sigma_1 \)
- See how long it takes to reject

Second byte is \( \sigma_1 \) that causes the longest response
Holiwudd Criptoe!

Most likely not what was meant by Hollywood, but conceivable
Thwarting Timing Attacks

Possibility:
• Use a string comparison that is guaranteed to take constant time
• Unfortunately, this is hard in practice, as optimized compilers could still try to shortcut the comparison

Possibility:
• Choose random block cipher key $k'$
• Compare by testing $F(k', A) == F(k', B)$
• Timing of “==” independent of how many bytes $A$ and $B$ share
Alternate security notions
Strongly Secure MACs

Output 1 iff:
• $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), \ldots\}$
• $\text{Ver}(k, m^*, \sigma^*) = 1$

$\text{SCMA-Adv}(\cdot) = \Pr[\text{outputs 1}]$
Strongly Secure MACs

Useful when you don’t want to allow the adversary to change *any* part of the communication.

If there is only a single valid tag for each message (such as in the PRF-based MAC), then (weak) security also implies strong security.

In general, though, strong security is stronger than weak security.
Adding Verification Queries

\[ k \leftarrow K \]
\[ \sigma_i \leftarrow \text{MAC}(k, m_i) \]
\[ b \leftarrow \text{Ver}(k, m, \sigma) \]

Output 1 iff:
- \( m^* \notin \{m_1, \ldots \} \)
- \( \text{Ver}(k, m^*, \sigma^*) = 1 \)

\[ \text{CMA}'-\text{Adv}(\cdot) = \text{Pr}[\text{ outputs 1}] \]
Theorem: \((\text{MAC,Ver})\) is strongly CMA secure if and only if it is strongly CMA’ secure
Improving efficiency
Limitations of CBC-MAC

Many block cipher evaluations

Sequential
Carter Wegman MAC

\[ k' = (k, h) \]

where:

- \( k \) is a PRF key for \( F: K \times R \rightarrow Y \)
- \( h \) is sampled from a pairwise independent function family

\[ \text{MAC}(k', m): \]

- Choose a random \( r \leftarrow R \)
- Set \( \sigma = (r, F(k, r) \oplus h(m)) \)
Theorem: If $F$ is $(t, q, \varepsilon)$-secure, then the Carter Wegman MAC is $(t-t', q-1, \varepsilon+1/|T|+q^2/|R|)$-strongly CMA secure.
Efficiency of CW MAC

$\text{MAC}(k',m)$:
- Choose a random $r \leftarrow \mathcal{R}$
- Set $\sigma = (r, F(k,r) \oplus h(m))$

$\mathcal{h}$ much more efficient than PRFs

PRF applied only to small nonce $r$
$\mathcal{h}$ applied to large message $m$
PMAC: A Parallel MAC

\[
F_{k'} \oplus k' \oplus F_{k'} \oplus k' \oplus F_{k'} \oplus k' \oplus F_{k'} \oplus k' \oplus F_{k'} \oplus k' \oplus F
\]

\[
\sigma = F_k
\]
Authenticated Encryption
Authenticated Encryption

Goal: Eve cannot learn nor change plaintext
  • Authenticated Encryption will satisfy two security properties
Syntax

Syntax:
• $\text{Enc}: K \times M \rightarrow C$
• $\text{Dec}: K \times C \rightarrow M \cup \{\bot\}$

Correctness:
• For all $k \in K$, $m \in M$, $\text{Dec}(k, \text{Enc}(k,m)) = m$
Unforgeability

\[
\begin{align*}
\text{Output 1 iff:} & \\
\bullet & c^* \notin \{c_1, \ldots\} \\
\bullet & \text{Dec}(k, c^*) \neq \bot
\end{align*}
\]
**Definition:** An encryption scheme \((\text{Enc}, \text{Dec})\) is an **authenticated encryption scheme** if it is unforgeable and CPA secure.
Constructing Authenticated Encryption

Three possible generic constructions:

1. MAC-then-Encrypt (SSL)

\[ k = (k_{Enc}, k_{MAC}) \]

\[ m \sigma \]

\[ MAC(k_{MAC}, m) \]

\[ Enc(k_{Enc}, (m, \sigma)) \]

\[ Dec(k_{Enc}, c) \]

\[ Ver(k_{MAC}, m, \sigma) \]

Accept

Reject
Constructing Authenticated Encryption

Three possible generic constructions:

2. Encrypt-then-MAC (IPsec)

\[
k = (k_{Enc}, k_{MAC})
\]

\[
m \xrightarrow{\text{Enc}(k_{Enc}, m)} c'
\]

\[
\sigma \xrightarrow{\text{MAC}(k_{MAC}, c')} c
\]
Constructing Authenticated Encryption

Three possible generic constructions:

3. Encrypt-and-MAC (SSH)

\[ k = (k_{Enc}, k_{MAC}) \]

\[ m \]

\[ \text{Enc}(k_{Enc}, m) \]

\[ \text{MAC}(k_{MAC}, m) \]

\[ \sigma \]

\[ c' \]

\[ c \]
Constructing Authenticated Encryption

1. MAC-then-Encrypt

2. Encrypt-then-MAC

3. Encrypt-and-MAC

Which one(s) \textit{always} provides authenticated encryption (assuming strongly secure MAC)?
Constructing Authenticated Encryption

MAC-then-Encrypt?
• Encryption not guaranteed to provide authentication
• May be able to modify ciphertext to create a new ciphertext
• Toy example: 
  \[ \text{Enc}(k,m) = (0, \text{Enc}'(k,m)) \]
  \[ \text{Dec}(k, (b,c)) = \text{Dec}'(k,c) \]
Constructing Authenticated Encryption

Encrypt-then-MAC?
• Inner encryption scheme guarantees secrecy, regardless of what MAC does
• (strongly secure) MAC provides integrity, regardless of what encryption scheme does

Theorem: Encrypt-then-MAC is an authenticated encryption scheme for any CPA-secure encryption scheme and *strongly* CMA-secure MAC
Constructing Authenticated Encryption

Encrypt-and-MAC?
• MAC not guaranteed to provide secrecy
• Even though message is encrypted, MAC may reveal info about message
• Toy example: \( MAC(k,m) = (m, MAC'(k,m)) \)
Constructing Authenticated Encryption

1. MAC-then-Encrypt ✘
2. Encrypt-then-MAC ✓
3. Encrypt-and-MAC ✘

Which one(s) always provides authenticated encryption (assuming strongly secure MAC)?
Constructing Authenticated Encryption

Just because MAC-then-Encrypt and Encrypt-and-MAC are insecure for some MACs/encryption schemes, they may be secure in some settings.

Ex: MAC-then-Encrypt with CTR or CBC encryption
• For CTR, any one-time MAC is actually sufficient
Theorem: MAC-then-Encrypt with any one-time MAC and CTR-mode encryption is an authenticated encryption scheme
Chosen Ciphertext Attacks
Chosen Ciphertext Attacks

Often, adversary can fool server into decrypting certain ciphertexts

Even if adversary only learns partial information (e.g. whether ciphertext decrypted successfully), can use info to decrypt entire message

Therefore, want security even if adversary can mount decryption queries
Chosen Plaintext Security

\[ \text{CPA-Exp}_{b}(\text{Charlie}, \lambda) \]
Chosen Ciphertext Security?

$$\text{Chosen Ciphertext Security?}$$

$$m_0^*, m_1^* \overset{\text{r}}{\leftarrow} \mathbb{K}$$

$$c^* \overset{\text{r}}{\leftarrow} \text{Enc}(k, m)$$

$$c \leftarrow \text{Enc}(k, m)$$

$$m \leftarrow \text{Dec}(k, c)$$

$$c^* \leftarrow \text{Enc}(k, m_b^*)$$
Lunch-time CCA (CCA1)

\[ k \leftarrow K \]
\[ c \leftarrow \text{Enc}(k, m) \]
\[ m \leftarrow \text{Dec}(k, c) \]
\[ c^* \leftarrow \text{Enc}(k, m_b^*) \]
Full CCA (CCA2)
Theorem: If \((\text{Enc}, \text{Dec})\) is an authenticated encryption scheme, then it is also CCA secure.
Proof Sketch

For any decryption query, two cases

1. Was the result of a CPA query
   • In this case, we know the answer already!

2. Was not the result of an encryption query
   • In this case, we have a ciphertext forgery
CCA vs Auth Enc

We know Auth Enc implies CCA security

What about the other direction?

For now, always strive for Authenticated Encryption
MAC-then-Encrypt with CBC

Even though MAC-then-Encrypt is secure for CBC encryption (which we did not prove), still hard to implement securely

Recall: need padding for CBC

Therefore, two possible sources of error
• Padding error
• MAC error

If possible to tell which, then Bleichenbacher attack
Using Same Key for Encrypt and MAC

Suppose we’re combining CBC encryption and CBC-MAC

Can I use the same key for both?
Attack?
Using Same Key for Encrypt and MAC

In general, do not use same key for multiple purposes
• Schemes may interact poorly when using the same key

However, some modes of operation do allow same key to be used for both authentication and encryption
CCM Mode

CCM = Counter Mode with CBC-MAC in Authenticate-then-Encrypt combination

Possible to show that using same key for authentication and encryption still provides security
Efficiency

So far, all modes seen require two block cipher operations per block
  • 1 for encryption
  • 1 for authentication

Ideally, would have only 1 block cipher op per block
OCB (Offset Codebook) Mode

\[
\Delta \leftarrow \text{Init}(N) \\
\Delta \leftarrow \text{Inc}_1(\Delta) \\
\Delta \leftarrow \text{Inc}_2(\Delta) \\
\Delta \leftarrow \text{Inc}_3(\Delta) \\
\Delta \leftarrow \text{Inc}_4(\Delta)
\]

\[
M_1 \rightarrow E_K \rightarrow C_1 \\
M_2 \rightarrow E_K \rightarrow C_2 \\
M_3 \rightarrow E_K \rightarrow C_3 \\
M_4 \rightarrow E_K \rightarrow C_4
\]

\[
\Delta \leftarrow \text{Inc}_5(\Delta) \\
\Delta \leftarrow \text{Checksum} \\
E_K \rightarrow \text{Final} \rightarrow \text{Tag} \\
\text{Auth} \\
\text{T}
\]

\[
\tau
\]
OCB Mode

 Twice as fast as other block cipher modes of operation

 However, not used much in practice
Other Modes

GCM: Roughly CTR mode then Carter-Wegman MAC

EAX: CTR mode then CMAC (variant of CBC-MAC)
After Spring Break

Hashing and commitment schemes

Public key cryptographic
• How to Alice and Bob exchange $k$ when over the internet?
Reminder

Homework 3 extension – due Tomorrow
Collision Resistant Hashing
Expanding Message Length for MACs

Suppose I have a MAC (MAC,Ver) that works for small messages (e.g. 256 bits)

How can I build a MAC that works for large messages?

One approach:
• MAC blockwise + extra steps to insure integrity
• Problem: extremely long tags
Hash Functions

Let \( h: \{0,1\}^n \rightarrow \{0,1\}^m \) be a function, \( m \ll n \)

\[
\text{MAC}'(k,m) = \text{MAC}(k, h(m))
\]
\[
\text{Ver}'(k,m,\sigma) = \text{Ver}(k, h(m), \sigma)
\]

Correctness is straightforward

Security?

- Pigeonhole principle: \( \exists m_0 \neq m_1 \text{ s.t. } h(m_0) = h(m_1) \)
- But, hopefully such collisions are hard to find
Collision Resistant Hashing?

Syntax:
• Domain $\mathbf{D}$ (typically $\{0,1\}^m$ or $\{0,1\}^*$)
• Range $\mathbf{R}$ (typically $\{0,1\}^n$)
• Function $\mathbf{H}: \mathbf{D} \rightarrow \mathbf{R}$

Correctness: $n << m$
Security?

Definition: \((\text{MAC,Ver})\) is \((t,\varepsilon)\)-collision resistant if, for all running in time at most \(t\),

\[
\Pr[H(x_0) = H(x_1) \land x_0 \neq x_1 : (x_0, x_1) \leftarrow \delta()]< \varepsilon
\]

Problem?
Theory vs Practice

In practice, the existence of an algorithm with a built in collision isn’t much of a concern
• Collisions are hard to find, after all

However, it presents a problem with our definitions
• So theorists change the definition
• Alternate def. will also be useful later
Collision Resistant Hashing

Syntax:
• Key space $K$ (typically $\{0,1\}^\lambda$)
• Domain $D$ (typically $\{0,1\}^m$ or $\{0,1\}^*$)
• Range $R$ (typically $\{0,1\}^n$)
• Function $H: K \times D \rightarrow R$

Correctness: $n \ll m$
Security

Definition: \( (\text{MAC,Ver}) \) is \((t,\varepsilon)\)-collision resistant if, for all running in time at most \(t\),

\[
\Pr[H(x_0) = H(x_1) \land x_0 \neq x_1: (x_0, x_1) \leftarrow (k), k \leftarrow K] < \varepsilon
\]
Collision Resistance and MACs

Let $h(m) = H(k,m)$ for a random choice of $k$

$MAC'(k_{MAC}, m) = MAC(k_{MAC}, h(m))$
$Ver'(k_{MAC}, m, \sigma) = Ver(k_{MAC}, h(m), \sigma)$

Think of $k$ as part of key for $MAC'$
Theorem: If \((\text{MAC,Ver})\) is CMA-secure and \(H\) is collision resistant, then so is \((\text{MAC}',\text{Ver}')\)
Proof

Hybrid 0

\[ \begin{align*}
    k_H &\leftarrow K_H \\
    k_{MAC} &\leftarrow K_{MAC} \\
    t_i &\leftarrow H(k_H, m_i) \\
    \sigma &\leftarrow \text{MAC}(k_{MAC}, t_i)
\end{align*} \]

Output 1 iff:
- \( m^* \notin \{m_1, \ldots \} \)
- \( \text{Ver}(k, t^*, \sigma^*) \) where
  \[ t^* \leftarrow H(k_H, m^*) \]
Proof

Hybrid 1

\[
\begin{align*}
\tau_i &\leftarrow H(k_H, m_i) \\
\sigma &\leftarrow MAC(k_{MAC}, t_i)
\end{align*}
\]

Output 1 iff:

- \( \tau^* \notin \{\tau_1, \ldots\} \)
- \( Ver(k, \tau^*, \sigma^*) \) where
  \[
  \tau^* \leftarrow H(k_H, m^*)
  \]
Proof

In Hybrid 1, negligible advantage using MAC security

If \( t^* \notin \{ t_1, \ldots \} \), then also forges
Proof

If succeeds in Hybrid 0 but not Hybrid 1, then
• $m^* \notin \{m_1, \ldots\}$
• But, $t^* \in \{t_1, \ldots\}$

Suppose $t^* = t_i$

Then $(m_i, m^*)$ is a collision for $H$