AUTHENTICATED ENCRYPTION

VIET TUNG HOANG
Agenda

1. AE and Its Security Definitions
2. Failed Ways to Build AE
3. Generic Compositions
4. Padding-Oracle Attack on SSL/TLS
So Far

Transfer $5 to account 12345

Privacy

Encryption scheme

Authenticated Encryption
Achieve both of these aims

Authenticity

MAC
Authenticated Encryption (AE)

Begin with two realizations

1. Authenticity is routinely needed/assumed
2. “Standard” privacy mechanisms don’t provide it

Provide an easier-to-correctly-use abstraction boundary
AE Syntax

Key Gen

\[ K \xrightarrow{\$} K \]

Encrypt

\[ M \xrightarrow{\$} C \]

Decrypt

\[ C \xrightarrow{\text{Decryption may reject invalid ciphertexts}} M \]

\[ K \]

\[ \mathcal{K} \]
Defining Security for AE

- Use Left-or-Right security for privacy

**Auth** $\mathcal{E}$

- **procedure Initialize()**
  - $K \leftarrow \$ \mathcal{K}$
  - Return $\mathcal{E}_K(M)$

- **procedure Enc($M$)**
  - Return $\mathcal{E}_K(M)$

- **procedure Finalize($C'$)**
  - Return $(\mathcal{D}_K(C') \neq \bot)$

**Adv**$_{\mathcal{T}}^{\text{auth}}(A) = \Pr[\text{Auth}^A_{\mathcal{E}} \Rightarrow 1]$
Agenda

1. AE and Its Security Definitions

2. Failed Ways to Build AE

3. Generic Compositions

4. Padding-Oracle Attack on SSL/TLS
Plain Encryption Doesn’t Provide Authenticity

**Question:** Does CBC provide authenticity?

**Answer:** No, because any ciphertext has valid decryption.
A Bad Fix: CBC with Redundancy

On decryption, verify the decrypted last block is zero.

**Question**: Break the authenticity of this scheme with a single Enc query.
An Attack

C₀  C₁  C₂  C₃

Enc

M₁  0^n

A

C₀  C₁  C₂

C₀

M₁

0^n

0^n

E⁻¹

E⁻¹

C₁

C₂

E_K

E_K

E_K

C₁

C₂

M₁

0^n
Complex Redundancy Doesn’t Help

Some (unkeyed) “redundancy” function, such as checksum

The redundancy is verified upon decryption

**Question**: Break the authenticity of this scheme with a single Enc query
An Attack

$C_0 \quad C_1 \quad C_2 \quad C_3$

$\text{Enc}$

$M$

$M_1 \quad h(M_1)$

$A$

$C_0 \quad C_1 \quad C_2$

$E_K$

$E_K$

$E_K$

$E_K^{-1}$

$E_K^{-1}$

$C_1 \quad C_2 \quad C_3$

$M_1 \quad h(M_1)$

$C_0$
A Case Study: WEP

Used in IEEE WiFi standard

24-bit IV is a part of the ciphertext
Attack 1: Exploiting Short IV

Assume all messages are of the same length, and fairly long

Goal: recover at least one message
Attack 1: Exploiting Short IV

Aim for an IV collision

For 24-bit IV’s, how many ctx to wait for collision prob ≈ 0.5?
Attack 1: Exploiting Short IV

Same IV, can recover $M_1 \oplus M_2$
**Attack 2: Chop-Chop Attack**

**Goal:** recover the underlying message by exploiting Dec queries
Attack 2: Chop-Chop Attack
Illustrated Via A Simpler Variant of WEP

Example:  Parity(10011) = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 1
Attack 2: Chop-Chop Attack
Illustrated For 4-bit Message
Decryption In CloseUp

\[ C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4 \rightarrow C_5 \]

\[ \text{IV} \rightarrow K \rightarrow \text{RC4} \rightarrow \text{Parity}(M) \rightarrow M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_4 \]

Compare with \( \text{Parity}(M_1 M_2 M_3) \)
Exploit Decryption Response

If valid, \( M_1 \oplus M_2 \oplus M_3 \oplus M_4 = 0 \)
Exploit Decryption Response

If invalid, \( M_1 \oplus M_2 \oplus M_3 \oplus M_4 = 1 \)
Exploit Decryption Response

Learn $M_1 \oplus M_2 \oplus M_3 \oplus M_4$
Exploit Decryption Even Further

Learn $M_1 \oplus M_2 \oplus M_3$
Solve A System of Linear Equations

\[
\begin{cases}
M_1 \oplus M_2 \oplus M_3 \oplus M_4 = \square \\
M_1 \oplus M_2 \oplus M_3 = \square \\
M_1 \oplus M_2 = \square \\
M_1 = \square
\end{cases}
\]
Agenda

1. AE and Its Security Definitions

2. Failed Ways to Build AE

3. Generic Compositions

4. Padding-Oracle Attack on SSL/TLS
Constructing AE: Generic Composition

A good PRF, such as Encrypted CBC-MAC

Privacy-only encryption (such as CTR/CBC)

Compose them to build AE

<table>
<thead>
<tr>
<th>Method</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt-and-MAC</td>
<td>SSH</td>
</tr>
<tr>
<td>MAC-then-Encrypt</td>
<td>SSL/TLS</td>
</tr>
<tr>
<td>Encrypt-then-MAC</td>
<td>IPSec</td>
</tr>
</tbody>
</table>
Encrypt-and-MAC: Simple Composition

No privacy: encrypting the same message results in the same tag
No authenticity if one can modify $C$ such that decryption is unchanged.
Encrypt-and-MAC in SSH

\[ M \]

Encode

\[ \text{len}(M) \| \text{len}(\text{pad}) \]

\[ M \quad \text{pad} \]

CBC

\[ C \]

Privacy | Authenticity
--- | ---
Yes | Yes
MAC-then-Encrypt

Privacy | Authenticity
--- | ---
Yes | No

for some bad encryption scheme

No authenticity if one can modify $C$ such that decryption is unchanged.
MAC-then-Encrypt in TLS

Privacy | Authenticity
---|---
Yes | Yes

\[ M \xrightarrow{F_{K_m}} \rightarrow M \xrightarrow{T} \rightarrow \text{CBC} \rightarrow C \]
Encrypt-then-MAC

\[ M \xrightarrow{\mathcal{E}_{K_e}} C \xrightarrow{F_{K_m}} T \]

<table>
<thead>
<tr>
<th>Privacy</th>
<th>Authenticity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Reusing Key May Lead to Attacks

EtM with CBC encryption and CBCMAC, same key

Good MAC if fixed input length

Break auth with one query
A Common Pitfall in Implementing EtM

Happened in ISO 1972 standard, and in RNCryptor of iOS

Forget to feed IV into MAC

Break auth with one query
1. AE and Its Security Definitions

2. Failed Ways to Build AE

3. Generic Compositions

4. Padding-Oracle Attack on SSL/TLS
The Padding-Oracle Attack

“Lucky Thirteen” attack snarfs cookies protected by SSL encryption
Exploit is the latest to subvert crypto used to secure Web transactions.

Meaner POODLE bug that bypasses TLS crypto bites 10 percent of websites
Some of the world's leading sites are vulnerable to an easier, more simplified attack.

Researchers poke hole in custom crypto built for Amazon Web Services
Even when engineers do everything by the book, secure crypto is still hard.

New TLS encryption-busting attack also impacts the newer TLS 1.3
Researchers discover yet another Bleichenbacher attack variation (yawn!).
Attack Model: Chosen Prefix Secret Suffix

Goal: Recover $M$
This Model Is Realistic: Attacking SSL

Encrypted communication via SSL

cookie with bank.com

visit

attacker.com
This Model Is Realistic: Attacking SSL

Request resource /AA at bank.com

Encrypted communication via SSL

Attacker.com

Cookie with bank.com

Bank of America

bank.com
This Model Is Realistic: Attacking SSL

$M$

encrypted

GET /AA cookie: $M$

Bank of America

bank.com

attacker.com
This Model Is Realistic: Attacking SSL

Enc oracle

GET /AA cookie: $M$

encrypted

$C$

Dec oracle

Bank of America

bank.com
Encryption In SSL: MAC-then-Encrypt

\[ M \rightarrow \text{MAC} \rightarrow M \oplus T \rightarrow \text{padding} \rightarrow \text{CBC} \rightarrow C \]
Padding In SSL Encryption

Consider byte strings only

- 31 bytes, 1 byte
- 24 bytes, 7 bytes, 1 byte
- 16 bytes, 15 bytes, 1 byte

block length is 16 bytes
The Attack in Action
Illustration For Two-block Message

Aim: Recover the message byte by byte
Recover Last Byte of First Block

Enc

Empty string

$\mathcal{E}_K(M_1M_2)$
Recover Last Byte of **First Block**

![Diagram showing the process of recovering the last byte of the first block.](image)
CBC Decryption

\[ V = M_1 \oplus C_0 \oplus C_3 \]

To pass MAC check, want the last byte of \( V \) to be 15

Pass with prob \( \sim 1/256 \)
Exploit Decryption Output

If valid, last byte of $V = M_1 \oplus C_0 \oplus C_3$ is 15

Learn last byte of $M_1$
Exploit Decryption Output

After $t$ attempts, succeed with prob $\sim 1 - (1 - 1/256)^t$ times
Exercise: Recover Last Byte of Second Block

$E_K(M_1M_2)$

$C_0$ $C_1$ $C_2$ $C_3$ $C_4$

Empty string

A

How to query Dec?
Recover **Second Last Byte of First Block**

The diagram shows a process where $Enc$ is applied to the blocks $B_0, B_1, B_2, B_3, B_4$. The input to $Enc$ is $0^8 M_1 M_2$, resulting in the last byte of the first block of $0^8 M_1 M_2$ being recovered.
Querying Dec: A Wrong Approach

This is the tag position, but the last byte is overwritten
How To Query Dec

\[ \mathcal{E}_K(M_1 M_2) = C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4 \]

\[ \mathcal{E}_K(0^8 M_1 M_2) = B_0 \quad B_1 \quad B_2 \quad B_3 \quad B_4 \]
CBC Decryption

\[ V = M'_1 \oplus B_0 \oplus C_3 \]

Learn last byte of \( M'_1 \)

0^8 then 15 bytes of \( M_1 \)
Patching Via Different Padding

Secure if implement properly
Careless Implementation Leads To Attack

- **Secure** if return a single error signal
- **Broken** if tell what kind of error it is.

Diagram:

- $C$ → CBC Dec
- CBC Dec → $M$ → MAC
- Correct padding?
- $T$ → $T'$
Implementation Is Hard: **Timing Leakage**
How To Attack
Illustration For Two-block Message

\[ C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4 \]

\[ \mathcal{E}_K(M_1 M_2) \]
Recover Last Byte of Second Block

```
Enc

C0  C1  C2  C3  C4

Dec

C0  R  C2

A

random
```
CBC Decryption

\[ V = M_2 \oplus C_1 \oplus R \]

If \( V \) ends with a zero byte

Bad tag signal