Authenticated Encryption

Viet Tung Hoang
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

$\mathcal{K}$ → $K$

Encrypt

$M$ → $\mathcal{E}$ → $C$

Decrypt

$C$ → $\mathcal{D}$ → \{ $M$ or $\perp$ \}

Decryption may reject invalid ciphertexts
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 \( \mathcal{D}_K(C') \neq \bot \)

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

Authenticity

**Enc**

Must never receive from Enc
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

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

\[ M_1 \quad 0^n \]

\[ C_0 \quad C_1 \quad C_2 \]

\[ C_0 \quad M_1 \quad 0^n \quad 0^n \]

\[ E_K \quad E_K \quad E_K \]

\[ C_1 \quad C_2 \quad C_3 \]

\[ E_K^{-1} \quad E_K^{-1} \]

\[ C_1 \quad C_2 \]

\[ C_0 \quad M_1 \quad 0^n \]
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

\[ C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow \text{Enc} \rightarrow M \]

\[ M_1 \rightarrow h(M_1) \rightarrow C_0 \rightarrow C_1 \rightarrow C_2 \]

\[ E_K \rightarrow E_K \rightarrow E_K \rightarrow E^{-1}_K \rightarrow C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow M_1 \rightarrow h(M_1) \]
A Case Study: WEP

24-bit IV is a part of the ciphertext

Used in IEEE WiFi standard
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

\[ \text{Same IV, can recover } M_1 \oplus M_2 \]
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

\[ M \rightarrow \text{Parity}(M) \]

\[ \text{IV} \rightarrow \text{RC4} \rightarrow C \]

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

Compare with Parity(M1M2M3)
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

\[ \text{Learn } M_1 \oplus M_2 \oplus M_3 \oplus M_4 \]
Exploit Decryption Even Further

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

\[
\begin{align*}
M_1 \oplus M_2 \oplus M_3 \oplus M_4 &= \square \\
M_1 \oplus M_2 \oplus M_3 &= \square \\
M_1 \oplus M_2 &= \square \\
M_1 &= \square
\end{align*}
\]
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

- **Privacy**: No
- **Authenticity**: No

For some bad encryption scheme

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

\[ M \rightarrow \text{Encode} \rightarrow \text{CBC} \rightarrow C \]

\[ \text{Privacy} \quad \text{Authenticity} \]
\[ \begin{array}{cc}
\text{Yes} & \text{Yes} \\
\end{array} \]

\[ \text{len}(M) \parallel \text{len}(\text{pad}) \rightarrow \]

\[ M \quad \text{pad} \rightarrow \text{counter} \rightarrow F_{K_m} \rightarrow T \]
MAC-then-Encrypt

- $M$ is encrypted with $F_{K_m}$
- The result is $T$
- $T$ is then encrypted with $E_{K_e}$ to give $C$

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

For some bad encryption scheme, there is **no authenticity** if one can modify $C$ such that decryption is unchanged.
MAC-then-Encrypt in TLS

\[ M \rightarrow F_{K_m} \rightarrow M \rightarrow T \rightarrow CBC \rightarrow C \]

<table>
<thead>
<tr>
<th>Privacy</th>
<th>Authenticity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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
Agenda

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

visit

M

cookie with bank.com

attacker.com

bank.com
This Model Is Realistic: Attacking SSL

encrypted communication via SSL

Request resource
/AA at bank.com

cookie with bank.com

attacker.com

Bank of America
This Model Is Realistic: Attacking SSL

encrypted

GET /AA cookie: $M$

attacker.com

bank.com
This Model Is Realistic: Attacking SSL

\[ M \]

GET /AA  cookie: \( M \)

\[ \text{Enc oracle} \]

\[ \text{Dec oracle} \]

\[ \text{encrypted} \]

\[ \text{bank.com} \]
Encryption In SSL: MAC-then-Encrypt

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

Consider byte strings only

<table>
<thead>
<tr>
<th>31 bytes</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>24 bytes</th>
<th>7 bytes</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>arbitrary</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16 bytes</th>
<th>15 bytes</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>arbitrary</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>
The Attack in Action
Illustration For Two-block Message

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

\[ \mathcal{E}_K(M_1M_2) \]
Recover Last Byte of First Block

\[ C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4 \]
To pass MAC check, want the last byte of $V$ to be 15

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

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

If invalid, restart

Dec

$C_0 \ C_1 \ C_2 \ C_3 \ C_1$

A

If invalid, restart

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

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

\[ \mathcal{E}_K(M_1M_2) \]

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

Second last byte of $M_1$ is now the **last byte** of the first block of $0^8 M_1 M_2$
Querying Dec: A Wrong Approach

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

\[ \mathcal{E}_K(M_1 M_2) = \begin{array}{ccccc}
C_0 & C_1 & C_2 & C_3 & C_4 \\
\end{array} \]

\[ \mathcal{E}_K(0^8 M_1 M_2) = \begin{array}{ccccc}
B_0 & B_1 & B_2 & B_3 & B_4 \\
\end{array} \]
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.

Correct padding?

Diagram:
- $C$ → CBC Dec → $M$ → $T$ → MAC → $T'$
"Given final block not properly padded"

About 16,000 results (0.32 seconds)

Stack Overflow
https://stackoverflow.com › questions › given-final-bl...

**Given final block not properly padded - java**

Nov 8, 2011 — BadPaddingException: **Given final block not properly padded**. Such issues can arise if a bad key is used during decryption.

7 answers · Top answer: If you try to decrypt PKCS5-padded data with the wrong key, and then ...

- **Given final block not properly padded** exception - Stack Overflow  Apr 11, 2018
- **Given final block not properly padded**. AES Decryption - Stack ...  Nov 13, 2022
- **Given final block not properly padded**. Such issues can arise if ...  Jul 10, 2020
- "Get Key Failed: **Given final block not properly padded**" when I ...  Sep 22, 2021

More results from stackoverflow.com
Implementation Is Hard: **Timing Leakage**

Source: Canvel et al, CRYPTO 2003
How To Attack
Illustration For Two-block Message

Enc

$\mathcal{E}_K(M_1M_2)$

Empty string
Recover Last Byte of Second Block

Enc

Dec

$C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4$

$C_0 \quad R \quad C_2$

random
CBC Decryption

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

If \( V \) ends with a zero byte

Bad tag signal