Dancing on the Lip of the Volcano: Chosen Ciphertext Attacks on Apple iMessage

Christina Garman
Matthew Green
Gabriel Kaptchuk
Ian Miers
Michael Rushanan
iMessage

• Created in 2011
• 1 billion deployed devices
• 200,000 messages per second peak
• First major deployment of end-to-end encrypted chat
• Used in other things:
  – Handoff
  – Other undisclosed products
All encryption is not equal

- **NO ENCRYPTION**
  - Data is sent from Service Provider to User without encryption.

- **TRANSPORT ENCRYPTION**
  - Data is encrypted between Service Provider and User.

- **END-TO-END ENCRYPTION**
  - Data is encrypted from Service Provider to User and vice versa.

**Data Flow:**
- **Unencrypted Data:** Data is not encrypted.
- **Encrypted Data:** Data is encrypted.
Reducing iMessage Security

- **NO ENCRYPTION**
  - Service Provider
  - Unencrypted Data
  - User

- **TRANSPORT ENCRYPTION**
  - Service Provider
  - Encrypted Data
  - User

- **END-TO-END ENCRYPTION**
  - Service Provider
  - Encrypted Data
  - User
“Server” can be very complex and insecure ...
... And have skilled attackers
iMessage
iMessage

- RSA encryption of $K$
- Message, AES-CTR encrypted with $K$
- ECDSA Signature by sender
Identity Misbinding Attack

- RSA encryption of $K$
- Message, AES-CTR encrypted with $K$
- ECDSA Signature by sender

- RSA encryption of $K$
- Mutated Ciphertext
- ECDSA Signature by ATTACKER
Ciphertext malleability

1. RSA encryption of $K$ → Message, AES-CTR encrypted with $K$ → ECDSA Signature by sender

2. RSA encryption of $K$ → Flip bits in AES ciphertext

3. RSA encryption of $K$ → Mutated Ciphertext → ECDSA Signature by ATTACKER
Chosen Ciphertext Attack

- Attacker can query on any ciphertext but challenged one
- “Who would build such a system?”

The CCA indistinguishability experiment $\text{PubK}_{A,\Pi}(n)$:

1. $\text{Gen}(1^n)$ is run to obtain keys $(pk, sk)$.
2. Adversary $A$ is given $pk$ and access to a decryption oracle $\text{Dec}_{sk}(\cdot)$, outputs a pair of messages $m_0, m_1$ with $|m_0| = |m_1|$. (These messages must be in the plaintext space associated with $pk$.)
3. A random bit $b \leftarrow \{0, 1\}$ is chosen, and then the ciphertext $c \leftarrow \text{Enc}_{pk}(m_b)$ is computed and given to $A$.
4. $A$ can continue to interact with the decryption oracle, but may not request decryption of $c$ itself. Finally, $A$ outputs a bit $b'$.
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

From Modern Cryptography Katz and Lindell
“Format Oracles”

• Suppose instead of decrypting the message, the server tells us if it is valid?

• E.g. Is the message the right length

• Or if it is encoded/serialized incorrectly
Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS...

Serge Vaudenay
Swiss Federal Institute of Technology (EPFL)
Serge.Vaudenay@epfl.ch

Abstract. In many standards, e.g. SSL/TLS, IPSEC, WTLS, messages are first pre-formatted, then encrypted in CBC mode with a block cipher. Decryption needs to check if the format is valid. Validity of the format is easily leaked from communication protocols in a chosen ciphertext attack since the receiver usually sends an acknowledgment or an error message. This is a side channel. In this paper we show various ways to perform an efficient side channel attack. We discuss potential applications, extensions to other padding schemes and various ways to fix the problem.

HTTP ERROR 500

Problem accessing /openidm/config/ui/configuration. Reason:
javax.crypto.BadPaddingException: Data must start with a block boundary.

Caused by:
org.forgerock.json.jose.exceptions.JweDecryptionException:
at org.forgerock.json.jose.jwe.handlers.enc:
at org.forgerock.json.jose.jwe.EncryptedJwt:
at org.forgerock.jaspi.modules.session.jwt:
iMessage: No padding, No XML, etc.

- RSA encryption of $K$
- Message, AES-CTR encrypted with $K$
- ECDSA Signature by sender
Over the lip of the Volcano
Finding a new format oracle
RSA encryption of $K$

Message, AES-CTR encrypted with $K$

ECDSA Signature by sender

Exploit some format

RSA encryption of $K$

FLIP Bits

ECDSA Signature by ATTACKER

RSA encryption of $K$

Mutated Ciphertext

Check if mutated ciphertext decrypts and validates
iMessage Format: What’s in the box?

- Builds on a partial RE by Quarks Lab
- Ciphertext is a GZIP compressed binary plist
- Part of the message is put in the RSA ciphertext to save space
Countermeasure

- The sender ID is stored in the ciphertext
- Client rejects if internal sender ID does not match external ID
- Luckily, the ciphertext is malleable!!
GZIP: another catch

- HEADER + compressed message + CRC32
- CRC checksum of decompressed message
- Decompression fails if the checksum is wrong!
Fun with CRC32

• Interesting mathematical fact: \( \text{CRC}(a) \oplus \text{CRC}(b) = \text{CRC}(a \oplus b) \)

• (slightly different for non-zero IVs)
Correcting CRCs for bit flips

M:

⊕

CRC(B)

B:

CRC(M)

M':

CRC(B)
There's a catch

M:

CRC(M)

⊕

B:

CRC(B)

Decrypt decompress

M':

CRC(M) CRC(B)
There’s a catch

M:

\[ \oplus \]

B:

\[ \begin{array}{cccccc}
0 & 0 & 0 & 1 & 0
\end{array} \]

\[ \text{CRC}(M) \]

\[ \text{CRC}(B) \]

\[ \text{CRC}(B) \]

M’:

\[ \oplus \]

\[ \text{CRC}(M) \]

\[ \text{CRC}(B) \]
There’s a catch

M: [encrypted]

⊕

B: 0 0 0 1 0

CRC(M)

CRC(B)

M': [decrypted]

CRC(M) ⊕ CRC(B)
GZIP uses DEFLATE for compression.

DEFLATE is:
- Lempel–Ziv encoding for repeated strings.
- Huffman coding of the resulting stream.

Flipping a bit in a Huffman symbol MAY NOT flip the same bit in the decoded character.

<table>
<thead>
<tr>
<th>Huffman Symbol</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>01000101</td>
</tr>
<tr>
<td>H</td>
<td>01001000</td>
</tr>
<tr>
<td>I</td>
<td>01001001</td>
</tr>
</tbody>
</table>
A

Message

CRC(M)

Decompressed message M'

CRC(M) ⊕ CRC(A ⊕ B 0^4)

CRC32(M)

M:

1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0

M':

BCR(M) ⊕ CRC(A ⊕ B 0^4)
Message

CRC(M)

Decompressed message $M'$

$\oplus$ Compressed and encrypted message

$\oplus$ Compressed and encrypted message

$\oplus$ Decompressed message $M'$
Message

$\text{CRC}(M)$


Compress Encrypt

$\oplus$

$11001011011100100$

Compressed and encrypted message

Decrypt decompress

$M'$:

CRC$(M) \oplus CRC(H \oplus E \ 0^4)$

Decompressed message $M'$

$\text{CRC32}(M)$
Decompressed and encrypted message

Compressed and encrypted message

Decompressed message $M'$

Encrypt decompress

$M$: H E ? ? ?

$M'$: H D ? ? ?

$CRC(M)$

$CRC(M) \oplus CRC(0^1 \oplus D \ 0^3)$
Message

M:

H E L ? ?

Decrypt decompress

\[ \oplus \]

1 1 0 0 1 1 0 1 1 0 0 1 0 0

Compressed and encrypted message

 Decompressed message \( M' \)

CRC32(M)

CRC(M)

\[ \oplus \]

CRC(M) \( \oplus \) CRC(0^2 \( \oplus \) T \( \oplus \) 0^2)
Decompress and encrypt the message $M$ to get the compressed and encrypted message. Then, decrypt the compressed message $M'$ to get the decompressed message $M''$. The crc is computed over the decompressed message $M''$. The final encrypted message is $CRC(M) ⊕ CRC(0^3 L ⊕ K ⊕ 0)$. The decompressed message $M''$ can be verified.
Message

M:

H
E
L
L
O

CRC(M)

Decompressed
message M'

Encrypt

Compressed
and encrypted
message

1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0

CRC32(M)

Decrypt
decompress

M':

H
E
L
L
N

CRC(M) ⊕ CRC(0^4 O ⊕ N )

Decompressed
message M'
Compression oracle?
RSA encryption of $K$

Message, AES-CTR encrypted with $K$

ECDSA Signature by sender

---

GZIP oracle

---

RSA encryption of $K$

Mutated Ciphertext

ECDSA Signature by ATTACKER

---

Check if mutated ciphertext decrypts and validates
Observing the oracle

• We need to see if when a message is received, it decompresses successfully
• iMessage does not report errors to the sender
• Read receipts require someone to see the message
iMessage
Attachment messages

• Can see if message decompresses
• Requests block download response to hide message
• Can mutate message to point download request to attacker controlled server (e.g. i8loud.com)
Attachment message payload

• {'gv': '8', 'pv': 0, 'p': ['mailto:alice.jhuisi@gmail.com', 'mailto:jhuisiscratch@gmail.com'], 'gid': 'A9CD06B6-6198-4289-A2C1-678B4E43ED77', 't': 'u￼', 'v': '1', 'x': '<html><body><FILE name="04duck.png" width="480" height="673" datysize="489847" mime-type="image/png" uti-type="public.png" mmcs-owner="MAB49B97D4B303E44942B4D05829B4F68012E577BBF0242A03E714F4B3F9D69CD.C01USN00" mmcs-url="https://p10-content.icloud.com/MAB49B97D4B303E44942B4D05829B4F68012E577BBF0242A03E714F4B3F9D69CD.C01USN00" mmcs-signature-hex="01AB6ED842CC96A19C19D1CF3FECA0CB37CE17B07D" file-size="489847" decryption-key="00F49B0E7388F578592FBB1618052675079DE82F0ABDE4BD5C4B2F5AF1426061DC"/> [ OPTIONAL MESSAGE] </body></html>'}
Attack gets harder

• Attachment messages use a dynamic Huffman table which we don’t know
• We must recover the table
  – We basically have to edit known plaintext in the message
  – Variable length symbols, so we don’t know which decompressed byte we are affecting
  – Detect symbol edges (with high probability) with double bit flips
Complete Attack

- Get message
- Change sender ID
- Use CRCs guess and check for chosen ciphertext attack to:
  - Recover Huffman table
  - Read attachment key
- Decrypt attachment with recovered key
Real attack

• Requires $2^{18}$ oracle queries
• Long tail on message processing times with an upper bound of 1 second, average of 390 ms
• Takes 73 hours to execute attack (reducible to 35 hours via backtracking)
• Recovered 232 of 256 bits in the encryption key for the attachment
Simulating larger numbers of attacks

- Recovers all but 40 bits of the key for 34% of messages (brute force < 24 hour)
- Recovers all but 24 bits of key for 23% of messages (brute force < 1 hour)
Ideal world solutions:

• Use Axolotl/Signal
• Just use authenticated encryption
  – AES-GCM/OCB
  – Include an HMAC
• Breaks backward compatibility
• Hard to do with 1 billion deployed devices
Real World Mitigations
Without breaking existing devices

• Recommended backward-compatible mitigations
  – Prevent the identity misbinding attack by moving sender ID to non-malleable RSA-OAEP ciphertext
  – Prevent chosen ciphertext attack by blacklisting RSA-OAEP ciphertexts that fail to decrypt
• RSA blacklisting deployed in IOS 9.3+ and OSX 10.11.4+
• Took Apple 4 months and 30 engineers to deploy
• Released on March 21, 2016
Apple says the Founding Fathers would be ‘appalled’ with the Justice Dept. for iPhone fight

By Mark Berman  March 15

Protesters outside an Apple store in Boston last month. (Steven Senne/AP)

The feud between Apple and the Justice Department took another turn Tuesday, as the technology giant used a new court filing to say that the Founding Fathers “would be appalled” with the government’s stance.

Apple argued that if the government prevails, it could force companies to do a number of things they would not do in the interest of national security. It also claimed that the FBI’s request was out of line with decades of legal precedent.
This shaky edifice could crumble at any moment.