How to Abuse and Fix Authenticated Encryption Without Key Commitment

Ange Albertini¹, Thai Duong¹, Shay Gueron²,³, Stefan Kölbl¹, Atul Luykx¹, Sophie Schmieg¹

¹Security Engineering Research, Google
²University of Haifa
³Amazon
What is Key Commitment?
What is Key Commitment?

Plaintext → AEAD Encrypt → (Ciphertext, Tag)
What is Key Commitment?
What is Key Commitment?

Invalid

AEAD Decrypt

K

(Ciphertext, Tag)
What is Key Commitment?

Plaintext $\rightarrow$ AEAD Decrypt $\rightarrow$ (Ciphertext, Tag)

$K$
What is Key Commitment?

K' → AEAD Decrypt → (Ciphertext, Tag)
Contributions

- Explore vulnerable settings and products.
- Study practical ways to exploit lack of key commitment.
- Provide simple and efficient ways to add key commitment.
Case Study: Envelope Encryption
Case Study: Envelope Encryption

Envelope Encryption

- All major cloud service providers use envelope encryption.
- Encrypt data with symmetric key (DEK), and wrap DEK under multiple symmetric or asymmetric recipient keys (KEK).
Case Study: Envelope Encryption

\[ \text{AEAD-Encrypt}(K_{\text{DEK}}, \text{"plaintext"}) \]
Case Study: Envelope Encryption

AEAD-Encrypt(K_{DEK}, "plaintext")

Wrap(K_{A}, K_{DEK})

"plaintext"
Case Study: Envelope Encryption

AEAD-Encrypt(\(K_{DEK}\), "plaintext")

Wrap(\(K_A, K_{DEK}\))
Wrap(\(K_B, K_{DEK}\))
AEAD-Encrypt(\(K_{DEK}, \text{"plaintext"}\))
Case Study: Envelope Encryption

- \( K_{DEK^*} \)
- \( K_{DEK} \)
- "plaintext"
- \( K_A \)
- \( K_B \)

\[
\begin{align*}
\text{Wrap}(K_A, K_{DEK^*}) & \quad \text{Wrap}(K_B, K_{DEK}) & \quad \text{AEAD-Encrypt}(K_{DEK}, \text{"plaintext"})
\end{align*}
\]
Case Study: Envelope Encryption

User A

AEAD-Encrypt($K_{DEK} \text{, “plaintext”}$)

$K_{DEK}^*$ → AEAD Decrypt

“malicious plaintext”

User B

AEAD-Encrypt($K_{DEK} \text{, “plaintext”}$)

$K_{DEK}$ → AEAD Decrypt

“plaintext”
Case Study: Envelope Encryption

- Recipients receive **same** ciphertext.
- Might **falsely** assume that everyone decrypts to the same plaintext.
- Without a key-committing AEAD this is **not** true.
- AWS Encryption SDK was affected (< 2.0.0) and patched (CVE-2020-8897).
Practical Examples

- Key Rotation (see Paper)
- Subscribe with Google (see Paper)
- Facebook Message Franking (CRYPTO’18)
- Partitioning Oracle Attacks (USENIX’21)
- age file encryption (Mirco Stäuble, ETH Zurich)
Exploiting Lack of Key Commitment
Exploiting Lack of Key Commitment

Most commonly used AEADs are not key committing:

- AES-GCM, **AES-GCM-SIV**.
- ChaCha20-Poly1305.
- OCB3.
Exploiting Lack of Key Commitment

Constructing valid ciphertexts under multiple keys puts restrictions on plaintext:

- Include random blocks of data.
- Fixing bits in plaintext to specific values.
Exploiting Lack of Key Commitment

File formats have various restrictions:

- Starting sequences
- Headers
- Length fields
- ...
Exploiting Lack of Key Commitment

Can we still create meaningful plaintexts which are compliant with common file formats?

- Tooling supports 40+ formats, allows 270+ combinations, automated.
- Provide examples for PDF/PE, HTML/HTML...
- Our ePrint paper includes a PDF viewer :-)

Google
Dissection

Top file

```
0x < ! - - - - - - > < html > Hello
1x < lo World ! < / html >
2x \r \n < ! - - 
```

Bottom file

```
3x < a href = " http : / / w
4x < w w . e v i l . c o m " > Cli
5x < c k here ! < / a > < / h t
6x < m l > < ! - - 
```

Padding
Tag correction
<table>
<thead>
<tr>
<th>Hex</th>
<th>Decimal</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x</td>
<td>a2</td>
<td>ae</td>
<td>0f</td>
<td>b0</td>
<td>21</td>
<td>7b</td>
<td>96</td>
<td>71</td>
<td>6f</td>
<td>ff</td>
<td>96</td>
<td>73</td>
<td>4f</td>
<td>96</td>
</tr>
<tr>
<td>1x</td>
<td>b5</td>
<td>9f</td>
<td>0e</td>
<td>bd</td>
<td>c8</td>
<td>cd</td>
<td>2e</td>
<td>ab</td>
<td>9f</td>
<td>5f</td>
<td>4c</td>
<td>2b</td>
<td>1d</td>
<td>56</td>
</tr>
<tr>
<td>2x</td>
<td>c3</td>
<td>67</td>
<td>f7</td>
<td>35</td>
<td>0d</td>
<td>d4</td>
<td>75</td>
<td>a0</td>
<td>d5</td>
<td>be</td>
<td>e1</td>
<td>66</td>
<td>53</td>
<td>63</td>
</tr>
<tr>
<td>3x</td>
<td>24</td>
<td>34</td>
<td>ee</td>
<td>d2</td>
<td>da</td>
<td>23</td>
<td>70</td>
<td>66</td>
<td>ea</td>
<td>02</td>
<td>01</td>
<td>e8</td>
<td>b2</td>
<td>45</td>
</tr>
<tr>
<td>4x</td>
<td>7f</td>
<td>b8</td>
<td>0b</td>
<td>ef</td>
<td>f3</td>
<td>91</td>
<td>eb</td>
<td>5c</td>
<td>7a</td>
<td>21</td>
<td>52</td>
<td>f8</td>
<td>71</td>
<td>7a</td>
</tr>
<tr>
<td>5x</td>
<td>a5</td>
<td>41</td>
<td>82</td>
<td>b2</td>
<td>7e</td>
<td>43</td>
<td>b3</td>
<td>e3</td>
<td>13</td>
<td>09</td>
<td>9a</td>
<td>a9</td>
<td>b9</td>
<td>d8</td>
</tr>
<tr>
<td>6x</td>
<td>41</td>
<td>48</td>
<td>d0</td>
<td>ab</td>
<td>90</td>
<td>5f</td>
<td>6e</td>
<td>d4</td>
<td>2d</td>
<td>59</td>
<td>0d</td>
<td>a4</td>
<td>24</td>
<td>54</td>
</tr>
<tr>
<td>7x</td>
<td>85</td>
<td>39</td>
<td>a5</td>
<td>af</td>
<td>35</td>
<td>be</td>
<td>2c</td>
<td>db</td>
<td>dc</td>
<td>c1</td>
<td>07</td>
<td>bf</td>
<td>98</td>
<td>ce</td>
</tr>
</tbody>
</table>

Ambiguous ciphertext
Hello World!

<html>
  <body>
    <p>First plaintext</p>
  </body>
</html>
Dissection of the first plaintext

0x 7c fa a9 b5

2x \r \n ! - -

Prefix (commented out)

Top file

Suffix (commented out)

< ! - -

2x

Prefix (commented out)

Top file

Suffix (commented out)

Dissection of the first plaintext
Dissection of the second plaintext
Adding Key Commitment
Adding Key Commitment

How to address lack of key commitment?

- Use key committing scheme in the first place.
- In paper we analyze two solutions compatible with AEADs:
  - Padding fix
  - Generic Construction
- Efficient Schemes for Committing Authenticated Encryption (EUROCRYPT’22).
Conclusion

Takeaways:
● Lack of key commitment is an issue in real-world applications.
● AEADs should be explicit about providing this property or not.

Resources available:
● https://eprint.iacr.org/2020/1456
● https://github.com/corkami/mitra
● https://github.com/kste/keycommitment
Questions?