IV-FV Authenticated Encryption and Triplet-Robust Decryption

Shoichi Hirose\textsuperscript{1}, Yu Sasaki\textsuperscript{2} and Kan Yasuda\textsuperscript{2}

1: University of Fukui
2: NTT Secure Platform Laboratories
13/January/2015 @ ESC 2015
Conventional AE

Sender

$\text{Enc}$
$\rightarrow$
$T, C$

$\text{Enc}$

$\text{Enc}$

$\rightarrow$
$\text{communicated data}$

$\rightarrow$
$\text{Receiver}$

$IV, A, M$

$\rightarrow$
$IV, A, T, C$

$A$: must be the same value as Enc input \textbf{ok}

$IV$: usually the same value as Enc input \textbf{????}

• By modifying $IV$, can we do something better?
Our Proposal: IV-FV AE via Tag-Feedback

 Sender $\text{Enc}(IV, A, M)$

 Receiver $\text{Enc}(IV, A, T, C)$

• It enhances decryption security in the RUP setting.
• Suitable design choice for IV-FV AE is discussed.
• More advantages:
  Secret Message Num, nonce stealing, mass surveillance

-Tag-feedback

requires only 1 TBC call
Research Background
Conventional Authenticated Encryption

Formalization of conventional AE:

\((IV: \text{Initialization Vector including nonce})\)

Encryption

\( (IV, A, M) \rightarrow (IV, A, T, C) \)

Decryption

\( (IV, A, T, C) \rightarrow M \lor \perp \)
Decryption Security in Theory

Avoid leaking info about $M$ when decryption fails

Generic approach: Verify-then-Decrypt

Start decryption only if verification succeeds.

• It works well in theory.

• In practice, it requires 2-pass computation.
  ➢ Expensive computational cost
  ➢ Decryption cannot be processed online
  ➢ Message needs to be stored in a large memory
Decryption Security in Practice

• Using intermediate tag is a possible solution
  - Extra communication cost
  - Exponential tag size for exponential message size

• Releasing Unverified Plaintext (RUP)
  - Releasing decryption results before verification
  - Formalized by Andreeva et al. in Asiacrypt 2014
  - Adversary can access to decrypt-anyway oracle
Strength of Decrypt-Anyway Oracle

• Decrypt-anyway oracle leaks a lot of information.

• Example: GCM mode
  - Encryption: CTR-mode
  - Authentication: GHASH

Decrypt-anyway oracle returns correct $M$ as long as $IV$ and $C$ are correct.

$(IV, A', T', C)$ for any wrong $(A', T')$ leaks information about $M$. 

Copyright©2014 NTT corp. All Rights Reserved.
(Fully) Robust Decryption

Queries to decryption-anyway oracle does not leak any info about $M$, unless $(IV, A, T, C)$ has been observed from the encryption oracle before.

- satisfied by AEZ
- Theoretically best possible, but 2-pass
- We want weaker but more efficient security.
Tag-Robust Decryption

Queries to decryption-anyway oracle does not leak any info about $M$, unless $T$ has been observed from the encryption oracle before.

$IV, A, M \rightarrow Enc \rightarrow (IV, A, T, C)$

- Satisfies by APE

[Figure 8, PRIMATEs v1 Submission to the CAESAR Competition]
APE has several drawbacks

- Backward decryption
- Tags cannot be truncated
- Correctly decrypt $C$ without knowing $A$ and $IV$

[Figure 8, PRIMATEs v1 Submission to the CAESAR Competition]
IV-Robust Decryption is hard. Decrypt-anyway oracle with unobserved IV can be meaningful.

Example: Attacks against GCM-mode

1. Query \((IV, A, T, C)\) to decrypt-anyway oracle.
2. Keystream for the queried IV is leaked.
3. The adversary predicts \(C\) for \(Enc(IV, A, M)\).

\[
Enc \xrightarrow{(IV, A,*)} \text{predictable} \quad \text{Devil} \xleftarrow{(IV, A, T, C)} M \xrightarrow{\text{Decrypt anyway}} C \oplus M
\]
Nonce decoy by [Andreeeva et al, Asiacrypt14]

\[ IV' \leftarrow PRF_{K'}(IV), \]
\[(IV, IV', A, C, T) \leftarrow Enc_K(IV', A, M) \]

**Dummy-IV-Robust Decryption**

Queries to decryption-anyway oracle does not leak any info about \( M \), unless \( IV' \) has been observed from the encryption oracle before.

- 1 additional block cipher call to create \( IV' \)
- Extra communication cost to share \( IV' \)
Summary of Previous Problems

• Nonce decoy is the technique for AE with forward decryption. It cannot work with APE (backward decryption).

• Dummy-IV robustness and tag-robustness cannot be combined.

• Tag-robust decryption requires anyway backward decryption, which we want to avoid.
Our Proposals
Formalization and Goal

Encryption: \((IV, A, M) \rightarrow (FV, A, T, C)\)
Decryption: \((FV, A, T, C) \rightarrow M\)
Verification: \((FV, A, T, C) \rightarrow \perp / T\)

Triplet-Robust Decryption

Queries to decryption-anyway oracle does not leak any info about \(M\), unless \((FV, A, T)\) has been observed from the encryption oracle before.

Achieve this security with minimum efficiency loss
IV-FV AE via Tag-Feedback

Encryption

\[ \text{Tag-feedback} \]

\[ IV \rightarrow E_{K2} \rightarrow FV, T \rightarrow \mathcal{E}_{K1} \rightarrow C \rightarrow M \]

\[ \text{The same cost as nonce decoy} \]

Decryption

\[ IV \rightarrow D_{K1} \rightarrow FV, T \rightarrow \mathcal{D}_{K1} \rightarrow M \]

\[ \text{wrong } (FV, T) \rightarrow M \text{ is random} \]
Incorporating AD (1/3)

- To achieve triplet-robust decryption, A needs to affect $IV-FV$ conversion in decryption.
- Process A with keyed function, and then mix.
Incorporating AD (2/3)

• To achieve triplet-robust decryption, $A$ needs to affect $IV-FV$ conversion in decryption.
• Process $A$ with keyed function, and then mix.

Exhaustive Combination

|       | $x$ | $y$ | $x \oplus y$ | $x||y$ |
|-------|-----|-----|--------------|--------|
| $x$   |     |     |              |        |
| $y$   |     |     | ✓            |        |
| $x \oplus y$ |     |     | ✓            |        |
| $x||y$ |     | ✓   | ✓            |        |
Incorporating AD (3/3)

- Small tweak size
- No necessity of $A$ to compute $FV$
Conditions on TBC for Tag-Feedback

**Necessary**
- Tweak size $\geq$ Block size
- SPRP security (not PRP) up to the birthday bound

**High Priority**
- Tweak size $>$ Block size
  (domain separation in tweak space to reduce #keys)

**Low Priority (negligible for long $M$)**
- Computational cost of TBC
- Faster re-tweak than re-key

**Unnecessary**
- Beyond-birthday-bound security
- Arbitrary tweak size
Design Choices of TBC

Three major approaches to build TBC
1. Block-cipher + mode (e.g. XE, XEX)
2. Direct tweaking on Luby-Rackoff and its GFN
3. Design from scratch

For approach 1, XEX is the most suitable.
   - Large tweak size
   - SPRP up to birthday

For approach 2, Luby-Rackoff is hard to implement in practice. We do not take this approach.
3. Design from Scratch

Old designs HPC and Mercy were attacked.

<table>
<thead>
<tr>
<th>Design</th>
<th>Block size</th>
<th>Tweak size</th>
<th>Key size</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threefish-256</td>
<td>256</td>
<td>128</td>
<td>256</td>
<td>too small tweak</td>
</tr>
<tr>
<td>Threefish-512</td>
<td>512</td>
<td>128</td>
<td>512</td>
<td>too small tweak</td>
</tr>
<tr>
<td>Threefish-1024</td>
<td>1024</td>
<td>128</td>
<td>1024</td>
<td>too small tweak</td>
</tr>
<tr>
<td>Deoxys-BC-128-128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>Deoxys-BC-256-128</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>Joltik-BC-128</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Joltik-BC-128</td>
<td>64</td>
<td>48</td>
<td>80</td>
<td>too small tweak</td>
</tr>
<tr>
<td>Joltik-BC-192</td>
<td>64</td>
<td>96</td>
<td>96</td>
<td>most preferable</td>
</tr>
<tr>
<td>Joltik-BC-192</td>
<td>64</td>
<td>64</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>KIASU-BC</td>
<td>128</td>
<td>64</td>
<td>128</td>
<td>too small tweak</td>
</tr>
<tr>
<td>SCREAM</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>iSCREAM</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>Silver</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td></td>
</tr>
</tbody>
</table>

Only Joltik-BC-192 satisfies: “tweak size > block size”
**Computation Order of Enc and Dec**

• Tag-feedback adds the TBC computation to the end of the underlying AE.
  - Enc can be online if underlying AE is online.
  - Dec can’t start until Enc generates $FV$.

• Computation order deserves careful attention.

• The situation is very different depending on applications:
  - Encrypted communication
  - Storage encryption
IV-FV AE in Encrypted Communication

• Computation power of Dec is often limited than Enc. → IV-FV AE can be “2-pass Enc, online Dec.”

   Enc: 1) Pre-compute $T$, then $FV$ offline
   2) Send $FV$ to Dec, then start to encrypt.
   Dec: 3) Start the decryption online

• This cost is different among EtM, E&M and MtE.
• E&M and MtE often computes $T$ very fast.
  \[ T \leftarrow E_K(\text{checksum}) \text{ in OCB} / \text{often processed by 4R AES.} \]
• EtM, needs encryption both of 1) and 2). **Bad!!**
IV-FV AE in Storage Encryption

• When Dec starts, $FV$ is already stored.
• Dec can be processed only in 1-pass.
  ➢ 1-pass online encryption
  ➢ 1-pass offline decryption
• 1-pass offline is better than 2-pass schemes.

Summary
  ➢ Communication mode: 2-pass Enc, 1-pass online Dec
  ➢ Storage mode: 1-pass online Enc, 1-pass offline Dec
Our Construction

• OCB3 as the conventional AE
  ➢ Tweakable block-cipher level
  ➢ Enc-and-MAC, suitable for communication usage

• Apply the tag-feedback
  ➢ We proved the triplet-robust decryption of our construction up to the birthday bound.
  ➢ Proof is omitted ......
Other Applications of IV-FV AE
Secret Message Number (1/2)

Introduced in the CAESAR call

• $IV$ protected w.r.t. integrity and confidentiality

Motivation of $SMN$ (Engineering origin)

• Suppose that $IV$ is the number of packets processed. Such sensitive information should not be exposed.

• To prevent the replay attack, protocols need to be stateful, e.g. embeds counter in $M$. But, counter shouldn’t be inside $M$, should be provided as $SMN$ as:
  ✓ to avoid the risk of wrong implementation
  ✓ to detect the replay fast
Tag-feed converts $IV$ to $FV$ with TBC.
- can support $SMN$
- (applying cipher to $IV$ seems the simplest way to support $SMN$)

$IV$ is recovered in the early stage of decryption.
- early detection of the replay.

6 schemes in CAESAR support $SMN$. $SMN$ is processed as the first block of $M$. (cost 1 block)
- Tag-feedback only requires the same cost.
Nonce Stealing in IV-FV AE

Proposed by Rogaway for processing $A$ efficiently.

- Mask is often an encryption of $IV$.
- #bits of security is often $b/2 \rightarrow |IV|$ can be $b/2$.
- $A$ can be processed in the extra space.

IV-FV AE can ensure the confidentiality of $IV$.

- $M$ or $SMN$ can be processed in the extra space.

Original

$IV \downarrow b$

$\downarrow E_K$

$\downarrow Mask$

Nonce Stealing

$IV \| |A \downarrow b$

$\downarrow E_K$

$\downarrow Mask$

Nonce Stealing in IV-FV AE

$\downarrow b$

$\downarrow Mask$

$SMN \| |M$

$\downarrow E_K$

$\downarrow Mask$
Mass Surveillance [Bellare et al. CRYPTO14]

\[ (K') \approx \text{System} \]

\[ IV \leftarrow E_{K'}(K) \]

\[ C, T \leftarrow Enc_K(IV, A, M) \]

\[ (K, A, M) \]

\[ (IV, A, C, T) \]

\[ (IV, A, C, T) \]

- If \( IV \) looks random, \( K' \) can be embedded in \( IV \).
  - Alice and Bob can communicate as usual with \( K \).
  - Malicious system can decrypt everything later.
- Can be prevented if \( IV \) is stateful e.g. counter.
  - Tag-feedback can easily handle state information.
Concluding Remarks
Concluding Remarks

• Proposed IV-FV AE framework
• Proposed generic conversion with tag-feedback
• Showed computation order for commu./storage
  ➢ E&M, MtE are more advantageous for commu.
• With 1 TBC call:
  ➢ Triplet-robust decryption in the RUP setting
  ➢ can support SMN, prevent replay attack (stateful)
  ➢ can absorb $M$ with nonce stealing
  ➢ can prevent mass surveillance (stateful)
Encryption

\[ IV \xrightarrow{E_{K2}} FV \]
\[ T \xrightarrow{E_{K1}} M \]
\[ C \]

Decryption

\[ M \xrightarrow{D_{K1}} IV \]
\[ FV \]
\[ T \]
\[ C \]

Thank you for your attention!!