Automated Analysis and Synthesis of Authenticated Encryption Schemes

Viet Tung Hoang  
University of Maryland  
Georgetown University

Jonathan Katz  
University of Maryland

Alex J. Malozemoff  
University of Maryland

ACM CCS 2015  
October 13, 2015
Authenticated Encryption (AE)

Authenticated Encryption: Achieve both of these aims

Privacy

Encryption scheme

Sender $K$  

Receiver $K$

Authenticity

Message Authentication Code (MAC)
Authenticated Encryption (AE)

- Lots of AE schemes
  - OCB, CCM, CAESAR candidates, etc.
Authenticated Encryption (AE)

• Lots of AE schemes
  • **OCB**, CCM, CAESAR candidates, etc.

  ✓ Most efficient
  ✗ Patented
Authenticated Encryption (AE)

- Lots of AE schemes
  - OCB, CCM, CAESAR candidates, etc.

  ✓ Not patented
  ✗ Slower than OCB
Authenticated Encryption (AE)

- Lots of AE schemes
  - OCB, CCM, CAESAR candidates, etc.

  Ongoing competition
  for new AE standard
  → Active area of research
Authenticated Encryption (AE)

- Lots of AE schemes
  - OCB, CCM, CAESAR candidates, etc.

- Developing new AE schemes is hard
  - Complex, error-prone proofs

*More systematic* way to build secure AE schemes?
Our approach

Automatically **analyze** and **synthesize** AE schemes

Extend [MalozemoffKatzGreen14], which analyzed / synthesized *encryption modes of operation*

Captures many existing schemes: OCB, XCBC, COPA, OTR, CCM, etc.
Our approach

**Step 1:** View AE scheme as graphs

- **Nodes:** operation (e.g., $E_K$, $\oplus$)
- **Edges:** Intermediate values

**Step 2:** Construct type system for graphs

- Typed graphs $\Rightarrow$ secure AE scheme

**Step 3:** Synthesize AE schemes using type system
Outline of rest of talk

1. Template for AE schemes
2. Viewing AE schemes as graphs
3. Type system for graphs
4. Implementation + results
Template for AE schemes

AE scheme defined by three algorithms: Enc, Dec, Tag

$M_i \in \{0, 1\}^n$

$C_i \in \{0, 1\}^n$

$C_i \in \{0, 1\}^n$

$M_i \in \{0, 1\}^n$
Template for AE schemes

\[ \begin{align*}
M_1 & \xrightarrow{\text{Enc}} C_1 \\
M_2 & \xrightarrow{\text{Enc}} C_2 \\
M_3 & \xrightarrow{\text{Enc}} C_3 \\
& \xrightarrow{\text{Tag}} V \\
0^n & \xrightarrow{\text{Dec}} M_1 \\
0^n & \xrightarrow{\text{Dec}} M_2 \\
0^n & \xrightarrow{\text{Dec}} M_3 \\
& \xrightarrow{\text{Tag}} V' \\
\end{align*} \]

Note: Ignoring nonce/associated data

\[ \mathcal{E}_K(\cdot) \]
Restricted class of AE schemes

Consider AE schemes using *tweakable blockciphers (TBCs)*

(Standard) blockcipher:

\[ M \in \{0, 1\}^n \]

\[ E_K \]

\[ C \in \{0, 1\}^n \]

\[ E_K: \text{Permutation on } \{0, 1\}^n \]
Restricted class of AE schemes

Consider AE schemes using \textit{tweakable blockciphers (TBCs)}

\begin{equation*}
M \in \{0, 1\}^n
\end{equation*}

\begin{equation*}
C \in \{0, 1\}^n
\end{equation*}

Tweakable blockcipher:

\begin{equation*}
E_K^T \text{: Permutation on } \{0, 1\}^n
\end{equation*}
Example AE scheme using TBCs: OCB

\[ \Sigma = M_1 \oplus M_2 \oplus M_3 \oplus M_4 \]
Viewing AE schemes as graphs

Enc

Dec

Tag

Enc

Dec

Tag
Type system for graphs

Each **node** assigned **type** corresponding to “property” of node output

Type = \{\perp, $, 0, 1\}

- \(\perp\): “Arbitrary”
- $: “Random”
- 0 and 1: Used in authenticity check (ignore for this talk)
Type system for graphs

Constraints on how nodes can be typed, e.g.:

Can show:
Node type $\$ \Rightarrow$ node output is (pseudo)random
From typed graphs to secure AE schemes

Need to show *two* properties: **privacy** and **authenticity**

**Privacy:** \( \Pr[A^{\mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[A^{\mathcal{S}(\cdot)} \Rightarrow 1] \) is small

---

**Goal**
From typed graphs to secure AE schemes

**Authenticity:** \( \Pr[A^{E_K}(\cdot) \text{ forges}] \) is small

\[ \text{valid} \quad C = C_1 \cdots C_\ell V \]

\( C \) must not be output of some prior query to \( E_K(\cdot) \)

Decrypting \( C_1 \cdots C_\ell \) produces random tag \( V' \)

\[ \Rightarrow V' \neq V \Rightarrow \text{Not valid forgery} \]
Implementation

Implemented *analyzer* and *synthesizer* in OCaml

**Analyzer:**

\[(\text{Enc}, \text{Tag}) \rightarrow \text{Derive Dec from Enc} \rightarrow (\text{Enc}, \text{Dec}, \text{Tag}) \rightarrow \text{Priv + auth checks} \rightarrow \checkmark \text{ or } ?\]
Implementation

Implemented *analyzer* and *synthesizer* in OCaml

Synthesizer:

\[ m, \text{Tag} \rightarrow \text{Generate all } \textbf{Enc} \text{ graphs of size } m \]

\[ (\text{Enc}, \text{Tag}) \rightarrow \text{Analyzer} \rightarrow \checkmark \text{ or } ? \]
Synthesis results

Ran synthesizer for **Enc** graphs of size 12-16...

<table>
<thead>
<tr>
<th>Size</th>
<th># Secure</th>
<th># Optimal</th>
<th># Parallel</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>13</td>
<td>13</td>
<td>5</td>
<td>47 sec</td>
</tr>
<tr>
<td>13</td>
<td>142</td>
<td>0</td>
<td>0</td>
<td>4.3 min</td>
</tr>
<tr>
<td>14</td>
<td>582</td>
<td>171</td>
<td>5</td>
<td>24.2 min</td>
</tr>
<tr>
<td>15</td>
<td>2826</td>
<td>40</td>
<td>6</td>
<td>2.8 hours</td>
</tr>
<tr>
<td>16</td>
<td>3090</td>
<td>66</td>
<td>1</td>
<td>3 hours*</td>
</tr>
</tbody>
</table>

**Total** 6653 290 17

- **Optimal**: 1 TBC per message block
- **Parallel**: TBC calls can be parallelized

* = Stopped analysis after given time
Synthesis results

Ran synthesizer for **Enc** graphs of size 12-16...

<table>
<thead>
<tr>
<th>Size</th>
<th># Secure</th>
<th># Optimal</th>
<th># Parallel</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>13</td>
<td>13</td>
<td>5</td>
<td>47 sec</td>
</tr>
<tr>
<td>13</td>
<td>142</td>
<td>0</td>
<td>0</td>
<td>4.3 min</td>
</tr>
<tr>
<td>14</td>
<td>582</td>
<td>171</td>
<td>5</td>
<td>24.2 min</td>
</tr>
<tr>
<td>15</td>
<td>2826</td>
<td>40</td>
<td>6</td>
<td>2.8 hours</td>
</tr>
<tr>
<td>16</td>
<td>3090</td>
<td>66</td>
<td>1</td>
<td>3 hours*</td>
</tr>
<tr>
<td>Total</td>
<td>6653</td>
<td>290</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>

OCB is the only previously known AE scheme of size 12
Example synthesized schemes

\[ \Sigma = M_1 \oplus M_2 \oplus M_3 \oplus M_4 \]

**Scheme 1**

**Scheme 2**
Example synthesized schemes: performance

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Encryption (cpb)</th>
<th>Decryption (cpb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCB</td>
<td>0.71</td>
<td>0.76</td>
</tr>
<tr>
<td>1</td>
<td>0.72</td>
<td>0.75</td>
</tr>
<tr>
<td>2</td>
<td>0.71</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Synthesized novel schemes on par with OCB
Conclusion

- Developed system for automatically *analyzing* and *synthesizing* AE schemes
- Able to synthesize schemes *as efficient and parallelizable as OCB*
- More results (e.g., automated attack generation) in paper

*Full Version:* https://eprint.iacr.org/2015/624
*Code:* https://www.github.com/amaloz/ae-generator