Synthesis of OWF-based Encryption Schemes

Martin Gagné

Université Grenoble 1, CNRS, VERIMAG, FRANCE

joint work with
Gilles Bartes, Juan Manuel Crespo, Benjamen Grégoire, César Kunz, Yassine Lakhnech and Santiago Zanella-Béguelin

Submitted to POPL ’13
Use recent advances in automated proving to help discover and verify new constructions for encryption schemes:

- build a synthesizer that outputs encryption scheme candidates
- use logic to filter out uninvertible candidates and discover decryption algorithm
- automatically prove IND-CPA security
- test for IND-CCA security
Grammar for encryption algorithms:

\[ e ::= r \mid 0 \mid m \mid f(e) \mid H(e) \mid e \oplus e \mid e \Vert e \]

Our encryption scheme synthesizer:

- generates all possible encryption algorithms requiring \( n \) commands
- uses symbolic logic to eliminate trivially insecure encryption scheme
- uses similar logic to synthesize decryption algorithm
Deducibility logic rules:

\[
\begin{align*}
\frac{e \vdash e_1 \quad e \vdash e_2}{e \vdash e_1 \| e_2} & \quad \text{Conc} \\
\frac{e \vdash e_1}{e \vdash (e_1 \oplus e_2)} & \quad \text{Xor} \\
\frac{e \vdash e_1 \| e_2}{e \vdash e_i} & \quad \text{Proj}_i \\
\frac{e \vdash e'}{e \vdash f(e')} & \quad \text{f} \\
\frac{e \vdash f(e')}{e \vdash e'} & \quad \text{finv}
\end{align*}
\]

- trivially insecure if you can deduce either $r$ or $m$ from ciphertext using non-boxed rules
- discover decryption algorithm by deducing $m$ using all rules (including boxed)
Proof search analyzes goals of the form \((c, X, E)\).

Start with \((c, X, b = b')\) where \(c\) is expression for ciphertext, \(X\) is a list of all \(H(e)\) in \(c\)

A goal is solvable if

- \(E\) is \(b = b'\) and \(b\) does not appear in either \(c\) or \(X\). The probability of \(E\) occurring is \(1/2\).
- \(E\) of the form \(e \in Q_H\) and \(e\) has a uniform random substring of length \(p\). The probability of \(E\) occurring is bounded by \(|Q_H|/2^p\)
- \(E\) is of the form \(e \in Q_H, f(r_1 \| \ldots \| r_n)\) is a substring of \(c\) with all \(r_i\) random, and a non-empty subset \(R \subseteq \{r_1, \ldots, r_n\}\) can be deduced from \(e\). The probability of \(E\) occurring is bounded by the probability of partially inverting \(f\) on \(R\)
If goal \((c, X, E)\) not solvable, modify goal using following rules:

- **Optimistic Sampling**: if \(r\) random, \(r \oplus e\) sub-expression of \(c\) and \(r\) never used elsewhere, replace all instances of \(r \oplus e\) by \(r'\) random

- **Permutation**: if \(r\) random, \(x := f(r)\) and \(r\) never used again, replace by \(x := r'\) for \(r'\) random

- **Failure Event**: find sub-expression \(H(e)\) in \(c\), set \(c' = c\{r / H(e)\}\) and \(X' = X - H(e)\), and solve goals \((c', X', E)\) and \((c', X', e \in Q_H)\)

- **Eager Sampling**: remove \(H(e)\) from code of encryption algorithm if \(H(e)\) does not appear in \(c\)

If rule \(\ast\) is not used, we can use EasyCrypt [BGHZ11] to produce proof with exact security bounds.
Chosen-Ciphertext Security

So far, no strategy to automatically generate sequence of games. We instead prove a general criterion for plaintext awareness

\[ H_0(t_0) \quad H_1(t_1) \oplus t_0 \quad \ldots \quad H_n(t_n) \oplus t_{n-1} \]

\[ H_0(t_0) \text{ checkbits} \quad t_n \vdash r \text{ or } G(r) \quad t_n \| r \vdash m \]
Chosen-Ciphertext Security

So far, no strategy to automatically generate sequence of games. We instead prove a general criterion for plaintext awareness

\[ H_0(t_0) \quad H_1(t_1) \oplus t_0 \quad \ldots \quad H_n(t_n) \oplus t_{n-1} \]

\[ H_0(t_0) \text{ checkbits} \quad t_n \leftarrow r \text{ or } G(r) \quad t_n || r \leftarrow m \]

Limitations

- not tight
- unlikely to ever get general enough
- cannot work for IND-CCA schemes that are not plaintext aware
Experimental Results

- Our synthesizer can generate more than 100,000 candidate encryption schemes in a few hours.
- Close to 3,000 IND-CPA schemes, close to 2,000 IND-CCA.
- All the filters, IND-CPA proof and IND-CCA test take less than 10 minutes for all candidates.
Experimental Results

- Our synthesizer can generate more than 100,000 candidate encryption schemes in a few hours
- Close to 3,000 IND-CPA schemes, close to 2,000 IND-CCA
- All the filters, IND-CPA proof and IND-CCA test take less than 10 minutes for all candidates

<table>
<thead>
<tr>
<th>Size</th>
<th>Unfiltered</th>
<th>Filtered</th>
<th>CPA (PA)</th>
<th>Redundant CPA (PA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>15</td>
<td>1</td>
<td>1 (0)</td>
<td>7 (4)</td>
</tr>
<tr>
<td>3</td>
<td>211</td>
<td>17</td>
<td>16 (0)</td>
<td>122 (112)</td>
</tr>
<tr>
<td>4</td>
<td>22,856</td>
<td>1,818</td>
<td>1,178 (711)</td>
<td>15,606 (12,682)</td>
</tr>
<tr>
<td>5</td>
<td>85,910</td>
<td>2,203</td>
<td>1,653 (1,154)</td>
<td>24,305 (19,996)</td>
</tr>
</tbody>
</table>
Future Work

For current grammar:

- Further optimize synthesizer to increase number of candidates
- Prove (in)completeness of automatic semantic security prover
- Attempt to prove IND-CCA security directly with sequence of games

Longer term:

- Use similar technique to generate schemes for larger set of complexity assumptions (Diffie-Hellman, lattices, etc)
- Develop new methods for proving security of encryption schemes with more complex security games (IBE, ABE, etc)
Future Work

For current grammar:

- Further optimize synthesizer to increase number of candidates
- Prove (in)completeness of automatic semantic security prover
- Attempt to prove IND-CCA security directly with sequence of games

Longer term:

- Use similar technique to generate schemes for larger set of complexity assumptions (Diffie-Hellman, lattices, etc)
- Develop new methods for proving security of encryption schemes with more complex security games (IBE, ABE, etc)
- Synthesis of signature, symmetric encryption, etc...