Automated Analysis and Synthesis of Block-Cipher Modes of Operation

Alex J. Malozemoff\textsuperscript{1}  Jonathan Katz\textsuperscript{1}  Matthew D. Green\textsuperscript{2}

\textsuperscript{1}University of Maryland

\textsuperscript{2}Johns Hopkins University

Designing/proving crypto constructions is hard

Can we automate the design/proof using ideas from program synthesis?

Program Synthesis

- Automatically construct programs based on (small) set of rules
- Has been applied to crypto protocols (e.g., [AGHP12, BCG+13])

This Work: Apply program synthesis to modes of operation
Background: Modes of Operation

Block-Cipher (= PRP, $F_k$): Encrypts fixed-length message (e.g., AES)

Mode of Operation: encrypts arbitrary-length messages, using block-cipher as building block

Example: Cipher-Block Chaining (CBC) Mode
Background: Security of Modes of Operation

Want output of mode to look “random” to adversary ⇒ IND$-CPA

What is IND$-CPA?

Adversary $A$ has oracle access to either
- (World 1) a truly random function
- (World 2) the desired mode of operation

$A$ specifies messages to encrypt and receives resulting ciphertexts

$A$’s Goal: Decide whether in World 1 or World 2

Secure: $A$ cannot distinguish between worlds

Note: Explains why ECB mode (encrypt each message block by PRP) is insecure
Motivation

Lots of modes exist; some modes are complex

Each scheme requires separate security proof
  • proofs occasionally omitted, sometimes wrong!

**Question**: Can we automate the security analysis, synthesize new modes?

**Solution**: Construct framework for automatically proving modes of operation secure, use this to synthesize new modes
This Work

Model mode as directed, acyclic graph
- Nodes $\rightarrow$ atomic operations
  - E.g., XOR two values, apply PRP to value, etc.
- Edges $\rightarrow$ intermediate values

Each edge can be assigned labels
- Constraints restrict how edges can be labeled

Meta-Theorem: There exists a valid labeling $\implies$ mode is secure

Note: Our approach analyzes constant size graph, yet proves security on arbitrary (polynomial) length inputs
Several prior works look at automatically analyzing modes:

- **Gagné et al. [GLLSN09, GLLSN12]:**
  - Modes described in imperative language
  - Use *compositional Hoare logic* to analyze security
  - **Drawback:** Can only reason about encryption of messages of pre-specified length

- **Courant et al. [CEL07]:**
  - Use *type system* to analyze security of modes, among others
  - **Drawback:** Similar to above

Our approach works for arbitrary (polynomial) length messages
Mode of Operation: Formal Definition

Defined by two algorithms:

- **Init**(1^n) → (c_0, z_0)
- **Block**(m_i, z_{i-1}) → (c_i, z_i)

**Enc_k**(m = m_1∥⋯∥m_ℓ):

- Compute (c_0, z_0) ← **Init**(1^n)
- For i = 1, . . . , ℓ:
  Compute (c_i, z_i) ← **Block**(m_i, z_{i-1})
- Output c_0∥⋯∥c_ℓ
Viewing Modes as Graphs

$\begin{align*}
IV \rightarrow & \quad m_1 & \quad m_2 & \quad m_3 \\
& F_k & F_k & F_k \\
& \downarrow IV & \downarrow c_1 & \downarrow c_2 & \downarrow c_3
\end{align*}$

$\begin{align*}
\Rightarrow \quad \text{Init algorithm} \\
\text{START} \rightarrow & \quad M \\
& \neg \quad \text{XOR} \\
& \neg \quad \text{PRP} \\
& \neg \quad \text{DUP} \\
& \neg \quad \text{OUT} & \neg \quad \text{NEXTIV}
\end{align*}$
Edge Labels (simplified): Intuition

Recall: Edges denote intermediate values

Intuition: Labels should capture “properties” of intermediate value

- Does value look random to adversary?
- Can value be output as ciphertext?
  - Only “random-looking” values should be output
- Can value be input into block cipher?
  - Only unique values should be input into block cipher
- etc.

Goal: If values on edges into OUT nodes look random to adversary, then mode is secure
Each edge label is a tuple \((\text{type}, \text{flags})\):

- **type** \(\in\{\bot, R\}\): “Type” of intermediate value
  - \(\bot\): Adversarially controlled
  - \(R\): Random
- **flags** \(\in\{0, 1\}^2\): Bit-vector denoting whether edge can be input into \(\text{OUT}\) or \(\text{PRP}\)
  - Prevents values being both output as part of ciphertext and input to \(\text{PRP}\)
Edge Constraints (simplified)

Example constraints:

- **GENRAND**: Outgoing edge gets type $\mathbb{R}$, flags.PR = 1, flags.OUT = 1
- **M**: Outgoing edge gets type $\perp$, flags.PR = 0, flags.OUT = 0
- **PRP**: Ingoing edge must have type $\mathbb{R}$ and flags.PR = 1; Outgoing edge same as GENRAND
Meta-Theorem

Want to prove: There exists a valid labeling $\Rightarrow$ mode is secure

Proof (high level): By induction:

- $\mathcal{A}$ inputs $m = m_1 \parallel \ldots \parallel m_\ell$ to mode
- Let $G$ be connected graph containing one copy of $\text{Init}$ and $\ell$ copies of $\text{Block}$
- Consider assigning values to edges in topological order step-by-step
- $\text{OUT}$: set of values on ingoing edges to $\text{OUT}$ nodes in $G$
- Invariant: values in $\text{OUT}$ are uniformly random
  - $\Rightarrow \mathcal{A}$ cannot distinguish between worlds
  - $\Rightarrow$ Proving invariant proves theorem!
- Considering each instruction, prove invariant holds by induction
  - Need additional invariants to prove main invariant
  - Gets messy... see paper for details
Implementation

Implemented model checker + synthesizer in OCaml

Model Checker:
Checks whether an input mode is secure
- **Recall**: Valid labeling \(\Rightarrow\) mode is secure
- \(\Rightarrow\) Determining secure mode is a constraint-satisfaction problem
- \(\Rightarrow\) Can use SMT solver (e.g., Z3)!

Secure modes need to be decryptable!
- Implement algorithm to check decryptability of mode

Synthesizer:
Can simply iterate over all possible graphs!
- Use simple rules to reduce search space
Ran model checker for modes with \( \leq 10 \) instructions

<table>
<thead>
<tr>
<th># Instructions</th>
<th>Valid</th>
<th>Decryptable</th>
<th>Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>559</td>
<td>282</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>3544</td>
<td>1361</td>
<td>87</td>
</tr>
<tr>
<td>10</td>
<td>8862</td>
<td>2101</td>
<td>243</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>13015</td>
<td>3774</td>
<td>355</td>
</tr>
</tbody>
</table>

We are able to synthesize all standard (secure) modes
- E.g., CBC, OFB, CFB, CTR, PCBC

**Note:** Slightly different numbers than in proceedings version
Conclusion

Introduced method for reasoning about modes of operation

- Uses only “local” analysis of single block

Meta-theorem: Validly labeled mode is secure

- ⇒ Can use SMT solver to automatically prove modes secure

Future Work:

- Handle additional operations (field operations, etc)
- Combine with EasyCrypt for (1) further security assurances and (2) concrete security bounds
- Can similar approach work for message authentication codes (authenticity), authenticated encryption (confidentiality and authenticity), etc?
Any questions?

**E-mail:** amaloz@cs.umd.edu  
**URL:** https://www.cs.umd.edu/~amaloz  
**Code:** https://github.com/amaloz/modes-generator