Automated Analysis and Synthesis of Block-Cipher Modes of Operation

Alex J. Malozemoff[1] Jonathan Katz[1] Matthew D. Green[2]

[1]University of Maryland

[2]Johns Hopkins University

Presented at the IEEE Computer Security Foundations Symposium, Vienna, Austria, July 19–22, 2014.

Introduction

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])

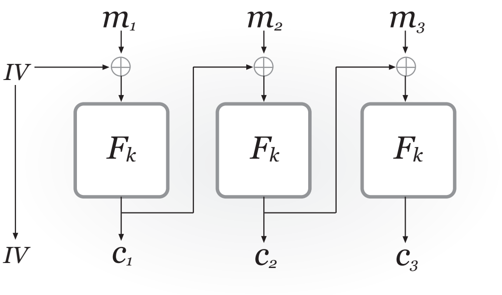
This Work: Apply program synthesis to modes of operation

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 $\Rightarrow$ IND\$-CPA

What is IND\$-CPA?

Adversary $\mathcal{A}$ has oracle access to either

- (World 1) a truly random function
- (World 2) the desired mode of operation

$\mathcal{A}$ specifies messages to encrypt and receives resulting ciphertexts

$\mathcal{A}$'s Goal: Decide whether in World 1 or World 2

Secure: $\mathcal{A}$ 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

Prior Work

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

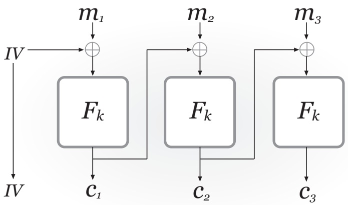
Mode of Operation: Formal Definition

Defined by two algorithms:

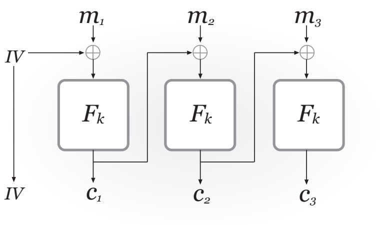
- $\mathsf{Init}(1^n) \rightarrow (c_0, z_0)$
- $\mathsf{Block}(m_i, z_{i-1}) \rightarrow (c_i, z_i)$

$\mathsf{Enc}_k(m = m_1 \| \cdots \| m_\ell)$:

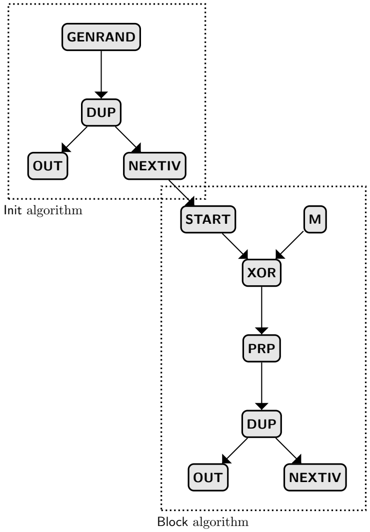
- Compute $(c_0, z_0) \leftarrow \mathsf{Init}(1^n)$
- For $i = 1, \ldots, \ell$:
  Compute $(c_i, z_i) \leftarrow \mathsf{Block}(m_i, z_{i-1})$
- Output $c_0 \| \cdots \| c_\ell$



Viewing Modes as Graphs



$\Rightarrow$



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.

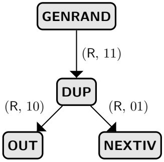
Goal: If values on edges into **OUT** nodes look random to adversary, then mode is secure

Edge Labels (simplified): Formalism

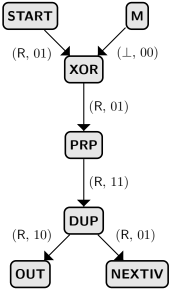
Each edge label is a tuple $(\mathbf{type}, \mathbf{flags})$:

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

Edge Constraints (simplified)



Init algorithm



Block algorithm

Example constraints:

- **GENRAND**: Outgoing edge gets type **R**, **flags.PRP = 1**, **flags.OUT = 1**
- **M**: Outgoing edge gets type $\perp$, **flags.PRP = 0**, **flags.OUT = 0**
- **PRP**: Ingoing edge must have type **R** and **flags.PRP = 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 $\mathbf{m} = \mathbf{m_1} \| \ldots \| \mathbf{m}_\ell$ to mode
- Let **G** be connected graph containing one copy of **Init** and ℓ copies of **Block**
- Consider assigning values to edges in topological order step-by-step
- *OUT*: set of values on ingoing edges to **OUT** nodes in **G**
- Invariant: values in *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

Results

Ran model checker for modes with $\leq$ **10** instructions

# Instructions	Valid	Decryptable	Secure
1–6	0	0	0
7	50	30	5
8	559	282	20
9	3544	1361	87
10	8862	2101	243
Total	13015	3774	**355**

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

- $\Rightarrow$ 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?

Thank You

Any questions?

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