Algorithmic Problems in Synthesized Cryptosystems (Extended Abstract)

Andrew M. Marshall [1] Catherine A. Meadows [2]
Paliath Narendran [3] Veena Ravishankar [1]
Brandon Rozek [1]

[1]University of Mary Washington, Fredericksburg, VA, USA

[2]Naval Research Laboratory, Washington, DC, USA

[3]University at Albany–SUNY, Albany, NY, USA

UNIF 2020

Initial Work

These result are **preliminary** work from a project looking at formal analysis in **synthesized crypotosystems**.
Idea:

- Want to generate secure cryptosystems automatically.
- Generally, cryptosystems are first generated, and then symbolic techniques are applied.
- Symbolic techniques used to identify secure cryptosystems and/or weed out insecure ones.

Goal:

- Create a tool which can generate and verify cryptosystems.
  - There will be restrictions of course.

Overview

- We analyze the synthesized cryptographic algorithms via a type of protocol modeling the interaction between an adversary and an encryptor/oracle.
- We denote these protocols as *Cryptographic Modes of Operation programs or MOO-programs*.

Security question: whether adversary can force the cryptosystem to produce cipher blocks that are equal modulo some theory.

Theory

In this paper we will primarily be concerned with the equational theory of *xor*:

- $E_{xor} = R_{\oplus} \cup E_{\oplus}$,
- $R_{\oplus} = \{x \oplus x \to 0,\ x \oplus 0 \to x\}$,
- $E_{\oplus} = AC(\oplus)$.

$\Sigma_{\oplus} = \{\oplus, f, 0\}$.

MOO Programs

MOO-Program: An interaction between the adversary and the oracle in which the adversary sends blocks of plaintext to be encrypted and the oracle sends back blocks of ciphertext according to some fixed *schedule* defined by the mode of operation.

MOO Programs

The blocks sent between the adversary and the oracle are modeled by terms:

- $MOO_{\oplus}$-terms are terms built up using the signature $\Sigma = \{\oplus, 0, f\}$.

A MOO-Program will be modeled by a list of $MOO_{\oplus}$-terms of the form $[t_1,\ t_2, \ldots,\ t_n]$.

An Example: CBC

One such example is Cipher Block Chaining (CBC) :

- The i^{th} plain text is a ground $MOO_{\oplus}$-term x_i.
- The i^{th} block of cipher text, C_i, is modeled by the term $f(C_{i-1} \oplus x_i)$, where x_i is the i^{th} plaintext.

The initial cipher block, the IV, is modeled by a bound variable.

An Example: CBC

MOO-Program models the CBC mode of encryption:

$$[IV, x_1, f(IV \oplus x_1), x_2, f(x_2 \oplus f(IV \oplus x_1))].$$

Detecting Possible Collisions

- We are interested in whether the problem will turn undecidable if we release the boundedness condition on either the number of sessions or their lengths.
- Finding whether adversary can force the cryptosystem to produce cipher blocks that are equal is decidable, when the number of sessions and their lengths are bounded.

Unbounded Session Lengths

Let us look at the non-deterministic version of the decision problem for MOO-programs of unbounded session lengths and bounded number of sessions is undecidable.

Encoding

We will use the following method for constructing cipher blocks. The construction encodes possible solutions to the Post Correspondence Problem (PCP):

- Let $PCP = \left(\frac{\alpha_0}{\beta_0}\right), \left(\frac{\alpha_1}{\beta_1}\right), \ldots, \left(\frac{\alpha_n}{\beta_n}\right)$.
- Let C_i be the i^{th} output of the cryptosystem.

The oracle will encode blocks as follows: For $i > 0$ let $C_i = E_{i_0}$ or, E_{i_2}, or $\ldots$, or E_{i_n} where,

- $E_{i_j} = [f(r_i \oplus C_{i,1}), f(r_i \oplus C_{i,2})],\ 0 \leq j \leq n,$
- $C_{i,1} = F(\alpha_j \bigoplus C_{i-1,1}),\ C_{i,2} = F(\beta_j \bigoplus C_{i-1,2}),$
- $C_{0,1} = F(\alpha_j \bigoplus 0),\ C_{0,2} = F(\beta_j \bigoplus 0).$

Reduction

Here is how the program works:

- The adversary non-deterministically picks a possible solution to the PCP, $i_0, i_1, i_2, \ldots, i_k$.
- Each turn the adversary sends an index in the solution, starting with i_k and proceeding each turn until i_0 is reached.
- At each step the oracle encodes a pair of cipher blocks E_j and returns them to the adversary.
- After receiving each E_j, the adversary attempts to check if any two cipher blocks are equal.

The program stops if the adversary finds two equal pairs or the adversary stops the program.

Theorem

Assume M is an arbitrary non-deterministic *MOO*-program. The problem of determining if M, executing with a bounded number of sessions and unbounded session lengths, ever produces two equal cipher blocks is undecidable.

Example

Consider the following PCP:

$$\overbrace{\left(\frac{ba}{baa}\right)}^{\text{block 1}}, \overbrace{\left(\frac{ab}{ba}\right)}^{\text{block 2}}, \overbrace{\left(\frac{aaa}{aa}\right)}^{\text{block 3}}$$

A solution to this problem is $1, 3$.

Example

Let's trace a run of the program where the adversary guesses the solution $1, 3$.
-In the first step the adversary sends 3 to the oracle and receives the following cipher block in return. $C_0 = E_{0_3}$ where

$$E_{0_3} = [f(r_0 \oplus C_{0,1}), f(r_0 \oplus C_{0,2})]$$

$$C_{0,1} = F(\alpha_3 \bigoplus C_{0,1}) = f(a \oplus f(a \oplus f(a \oplus 0)))$$

$$C_{0,2} = F(\beta_3 \bigoplus C_{0,2}) = f(a \oplus f(a \oplus 0))$$

Example

At the second step the adversary sends a 1 to the oracle and receives the following in return. $C_1 = E_{1_1}$ where

$$E_{1_1} = [f(r_1 \oplus C_{1,1}), f(r_1, C_{1,2})]$$

$$C_{1,1} = F(\alpha_1 \bigoplus C_{0,1}) = f(b \oplus f(a \oplus C_{0,1}))$$

$$C_{1,2} = F(\beta_1 \bigoplus C_{0,2}) = f(b \oplus f(a \oplus f(a \oplus C_{0,2})))$$

Notice that now after step 2 the adversary has two cipher blocks, $C_{1,1}$ and $C_{1,2}$, which are equal.

$$C_{1,1} = f(b \oplus f(a \oplus f(a \oplus f(a \oplus f(a \oplus 0)))))$$

$$C_{1,2} = f(b \oplus f(a \oplus f(a \oplus f(a \oplus f(a \oplus 0)))))$$

Other Cases?

Other undecidability results using similar reduction:

- deterministic unbounded session length,
- both the deterministic and non-deterministic form of unbounded number of sessions with bounded session length.

Invertibility

- A natural requirement of any cryptographic algorithm is that it be *invertible*.
- In the case of modes of encryption that leads to a question:
  - given a set S of *MOO* terms with subterms designated as plain text, can we tell if S is invertible?

Preliminaries

A few preliminaries:

- The set $\mathcal{C} = \{C_0, C_1, \ldots, C_n\}$ represents the cipher blocks, C_i, produced by the oracle in the *MOO*-program.
- Let $P = \{p_0, p_1, \ldots, p_n\}$ be the set representing the plaintext messages, which are constants, sent by the adversary, during a run of the *MOO*-program, where p_i is a subterm of C_i.

Invertibility Relation

- We can first define an *invertibility* relation, $\phi \vdash_E p$, s.t. $p \in P$.
- This relation can be axiomatized by a set of inference rules which introduce a new symbol, f^{-1}.
- $f = enc(\_, K)$ and $f^{-1} = dec(\_, K)$, s.t. $f^{-1}(f(p)) = p$

Invertibility Relation

In this case the set of rules axiomatizing the invertibility problem are exactly those axiomatizing the deduction problem [**?**, **?**].

While the deduction problem is undecidable in general it is decidable for some theories. Although the deduction problem is more general than the invertibility problem it does allows us to obtain a general decidability result for the theory of interest.

General Invertibility

- We limit our investigation to signature $\Sigma = \{\oplus, 0, f, f^{-1}\}$ and *MOO*-programs over this signature.
- The equational theories can be presented as a combination.
  $R_\oplus = \{x \oplus x \to 0,\ x \oplus 0 \to x\}$,
  $R_f = \{f(f^{-1}(x)) \to x,\ f^{-1}(f(x)) \to x\}$, $E_\oplus = AC(\oplus)$.
- We are interested in the invertibility problem for the theory $E^{-1} = R_f \cup R_\oplus \cup E_\oplus$.

General Invertibility

We can consider the combination problem for the theory $R_f \cup \{R_\oplus \cup E_\oplus\}$. Notice that this is a disjoint combination since $\{f, f^{-1}\} \cap \{0, \oplus\} = \emptyset$. Therefore from the disjoint combination result of [**?**] we get the following:

- The deduction problem is decidable in the theory $R_f \cup \{R_\oplus \cup E_\oplus\}$.

As a corollary we get:

- The invertibility problem for the theory $R_f \cup \{R_\oplus \cup E_\oplus\}$ is decidable.

More Efficient Algorithm

- Let $\mathcal{C} = \{C_o, C_1, \ldots, C_m\}$ be the set of cipher-blocks from a *MOO*-program.
- Let N denote any initial nonces, random strings, known by the adversary.
- The set of initial knowledge is $K = \mathcal{C} \cup N$. Let $S = \max\{|t|, s.t.\ t \in \mathcal{C}\}$.
- The set of *saturated* knowledge, K^*, is computed as follows:
  1. Initially $K^* = K$.
  2. Three closure operations are applied until there is no change to the set K^*:
     1. If $t \in K^*$, $t(\epsilon) = f$, $f^{-1}(t) \rightarrow t'$, then $K^* = K^* \cup \{t'\}$.
     2. If $t_1, t_2 \in K^*$, $t_1 \oplus t_2 = t'$, and $|t'| \leq S$ then $K^* = K^* \cup \{t'\}$.
     3. If $t \in K^*$, and $|f(t)| \leq S$ then $K^* = K^* \cup \{f(t)\}$.

Algorithm 1 Invertibility for $MOO_{\oplus}$ terms

Require: Set K, a plain-text goal p, and a set $K_p \subseteq K$ of terms containing p as a subterm.

if $K_p = \emptyset$ **then**

Exit with Failure.

else

Compute K^*

end if

if $p \in K^* \vee \exists t \in K^* \ s.t. \ t =_{R_f \cup R_{\oplus} \cup E_{\oplus}} p$ **then**

Return success.

else

Exit with Failure

end if

Result

Theorem 3.1

Algorithm **??** *is terminating, sound, and complete for the theory* $R_f \cup \{R_\oplus \cup E_\oplus\}$.

Example

- Let $C_0 = p_0 \oplus f(IV)$, $C_1 = p_1 \oplus f(p_0) \oplus f(IV)$, $C_3 = p_2 \oplus f(p_1) \oplus f(p_2)$.
- If IV is known, $IV \in N$, then $f(IV) \in K^*$ and from C_0 we obtain, p_0.
- Once we have p_0, $f(p_0) \in K^*$ and from C_1 we get p_1. Likewise, we can also obtain p_2. Thus we have invertibility.

Future Work

Future work includes:

- considering the invertibility problem for MOO program,
- identifying decidable cases of the decision problem,
- and implementing the decidable cases into a new tool for the automatic synthesis and security verification of certain cryptosystems.

Questions?

Thank you!