Sound Computational Interpretation of Formal Encryption with Composed Keys

Peeter Laud

peeter_l@ut.ee

Tartu University

Ricardo Corin

corin@cs.utwente.nl

University of Twente

Two views of cryptography

- To analyse cryptographic protocols, we must abstract them somehow.
  - Abstract messages, communication, possible operations, etc.
- We also have to abstract the adversary's capabilities.
- The security requirements have to be specified in the language of abstraction.
- Over years, two different abstractions (views) have evolved.

Formal view

- Messages are elements of a certain term algebra.
- Operations correspond to construction and destruction of these terms.
  - There are certain preconditions on when a message can be taken apart.
- The adversary has the same capabilities.
- Confidentiality of a certain message means the adversary's inability to obtain the term corresponding to that message.
- Quite different from real world, but automatic security analyses are possible.

Computational view

- Messages are bit-strings, operations work with bit-strings.
  - Certain operations (encryption) may be probabilistic.
- Adversary may be any efficient algorithm.
- Confidentiality of a message means its independence of the adversary's view.
  - Defined through the notion of indistinguishability of (probability distributions over) bit-strings.
- Close to real world, but security proofs have had to be hand-crafted.

Reconciling these two views

- These two views have developed quite independently.
- Only recently, results formally connecting these two views have started to appear.
  - First results date to 2000.
- Shape of these results — if some protocol or part of it is secure in the formal view then it is also secure in the computational view.
  - Results may handle a subset of the language. . .
  - as well as a subset of the attacks.

Still very much a work in progress.

Shape of some reconciling theorems

- Define the set of objects from the formal world.
  - Could be formal messages, protocols of certain shape, etc.
- Define the computational interpretation of these objects.
  - The interpretation belongs to the computational world.
    - Formal messages $\longrightarrow$ (probability distributions over) bit-strings.
    - Protocols $\longrightarrow$ set of interacting machines (compute with bit-strings; may be probabilistic).
- Define an equivalence relation over the formal objects.
- Show that the computational interpretations of equivalent objects are indistinguishable.

Abadi/Rogaway's original result

- Formal messages E are

$$\begin{array}{rcl} E & ::= & K \\ & | & C \\ & | & (E_1, E_2) \\ & | & \{E\}_K \end{array}$$

  where $K \in \mathbf{Keys}$ and $C \in \mathbf{Consts}$.

- Computational interpretation — the natural one.
  - Secure encryption system for interpreting formal encryption and keys.
  - All bit-strings tagged with their types.
- Equivalence of formal messages — see [3].

Our contribution

- Formal messages E are

$$\begin{array}{rcl} E & ::= & K \\ & | & C \\ & | & (E_1, E_2) \\ & | & \{E_1\}_{E_2} \end{array}$$

  i.e. we allow encryption keys to be arbitrary expressions.

- Interpretation of encryption:

$$[\![\{E_1\}_{E_2}]\!] = \mathcal{E}(R([\![E_2]\!]), [\![E_1]\!])$$

  $\mathcal{E}$ — secure encryption. R — random oracle.

Why composed keys matter?

- It is common to create new session keys from shared secrets.
- The adversary might create messages by encrypting with arbitrary data.
  - If the adversary were active.
    - We only consider formal messages, therefore only passive adversaries.
  - Consider possible future work.

Using the random oracle

- The random oracle returns a new, randomly generated bit-string each time it is queried with a new argument.
- In practice, it is replaced by some “random-looking” function.
  - For example, RSA-OAEP uses functions based on SHA-1.
- We use random oracle to guarantee the goodness and independence of encryption keys.
- Using the random oracle might make our result seem “impure”.
- However, we know of no results where the adversary is allowed to have some idea about the randomness used at key generation.

Patterns

- Abadi and Rogaway define $E_1 \cong E_2$ as follows:
  - For a formal expression E, define the set keys $\mathbf{K}$ that occur in E but that the adversary cannot find.
  - Replace subexpressions $\{\cdots\}_K$ of E, where $K \in \mathbf{K}$, by $\square$ ("the undecryptable").
    - This gives the pattern of E.
    - Denote it by $p(E)$.
  - $E_1 \cong E_2$ if $p(E_1) =_\alpha p(E_2)$.

On encryption cycles

- There is a definite gap between formal and computational worlds.
- Key K_1 encrypts key K_2 in expression E, if some $\{\cdots K_2 \cdots\}_{K_1}$ is a subexpression of E.
  - Here K_2 occurs not only as the encryption key.
- An encryption cycle is a cycle of the relation encrypts.
  - Example: $(\{K_2\}_{K_1}, \{K_1\}_{K_2})$.
- Encryption cycles are secure in formal world, insecure in computational.
- Abadi and Rogaway define their equivalence relation only for expressions without encryption cycles.

Encryption cycles and composed keys

- What are the encryption cycles with composed keys?
  - The occurrences of different parts of the complex key elsewhere in the expression should somehow be accounted for.
- We cannot define them. Hence we define $E_1 \cong E_2$ for all expressions.
- We also define it through the notion of pattern.
- Abadi and Rogaway defined the transformation from E to $p(E)$ as a one-step process.
- We define it in several steps, replacing $\{\cdots\}_E$ by $\Box$ one expression-occurring-as-key E a time.
  - After each replacement we look for a new suitable E.

Conclusions

- Our work is a step in the reconciliation of two views of cryptography.
- It showed some possible new directions of research.
  - Encryption systems, where the adversary has some information on or can somewhat influence the key generation.