EasyCrypt - CodeCraftLab

- Mastering the Art of EasyCrypt Programming -

Ji, Yong-Hyeon

A document presented for the EasyCrypt

Department of Information Security, Cryptology, and Mathematics
College of Science and Technology
Kookmin University

December 24, 2024

Contents

Symbols

In this paper, symbols are defined as follows.

$I \models P$	I satisfies P
$I \not\models P$	I does not satisfies P

Chapter 1

Mathematical Background

Lemma

For any boolean values a and b:

$$\neg(a \lor b) \Leftrightarrow (\neg a \land \neg b)$$

Proof.

We need to show both directions of the equivalence:

$$\neg(a \lor b) \Rightarrow (\neg a \land \neg b) \quad \text{and} \quad (\neg a \land \neg b) \Rightarrow \neg(a \lor b)$$

First Direction: $\neg(a \lor b) \Rightarrow (\neg a \land \neg b)$
1. Assume $\neg(a \lor b)$.
2. To show $\neg a$:

- **Case** $a = \text{true}$:

$$a = \text{true} \Rightarrow a \lor b = \text{true} \Rightarrow \text{contradiction with } \neg(a \lor b).$$

- Therefore, $\neg a$.

3. To show $\neg b$:

- **Case** $b = \text{true}$:

$$b = \text{true} \Rightarrow a \lor b = \text{true} \Rightarrow \text{contradiction with } \neg(a \lor b).$$

- Therefore, $\neg b$.

4. We conclude that $\neg(a \lor b) \Rightarrow (\neg a \land \neg b)$.
Second Direction: $(\neg a \land \neg b) \Rightarrow \neg(a \lor b)$
1. Assume $\neg a \land \neg b$.
2. Consider $a \lor b$:

- **Case** $a = \text{true}$: Contradicts $\neg a$.

- **Case** $b = \text{true}$: Contradicts $\neg b$.

3. Therefore, $(\neg a \land \neg b) \Rightarrow \neg(a \lor b)$.
Conclusion:

$$\neg(a \lor b) \Leftrightarrow (\neg a \land \neg b)$$

textbfLemma

For any boolean values a and b:

$$\neg(a \lor b) \Leftrightarrow (\neg a \land \neg b)$$

Proof.

1. To prove: $\neg(a \lor b) \Rightarrow (\neg a \land \neg b)$
   Assume: $\neg(a \lor b)$
   Then, we have:
   $a = \text{true} \Rightarrow a \lor b = \text{true} \Rightarrow \bot \Rightarrow \neg a$
   $b = \text{true} \Rightarrow a \lor b = \text{true} \Rightarrow \bot \Rightarrow \neg b$
   $\therefore \neg a \land \neg b$

2. To prove: $(\neg a \land \neg b) \Rightarrow \neg(a \lor b)$
   Assume: $\neg a \land \neg b$
   Then, $a \lor b = \text{true} \Rightarrow a = \text{true} \lor b = \text{true} \Rightarrow \bot$
   $\therefore \neg(a \lor b)$

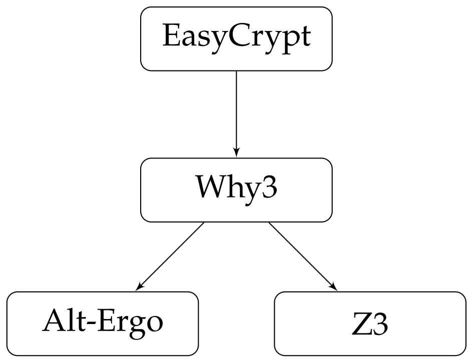
Conclusion: $\neg(a \lor b) \Leftrightarrow (\neg a \land \neg b)$

Chapter 2

Cryptographic Background

2.1 Attacker Type

- COA; Ciphertext-Only Attack
- KPA; Known-Plaintext Attack
- CPA; Chosen-Plaintext Attack
- CCA; Chosen-Ciphertext Attack
- CCA2; Adaptive Chosen-Ciphertext Attack



Integration of Why3, Alt-Ergo, and Z3 within Easy-Crypt

- $\mathcal{K}$: key space
- $\mathcal{N}$: nonce space
- $\mathcal{P}$: plaintext space
- $\mathcal{C}$: ciphertext space

$$\texttt{used\_once}(n, \mathcal{N}) = n \in \mathcal{N}.$$

$$\begin{array}{rcrcl} \texttt{used\_once} & : & \mathcal{N} \times \{\mathcal{N}\} & \longrightarrow & \{0,1\} \\ & & (n, \mathcal{N}) & \longmapsto & n \in \mathcal{N} \end{array}$$

$$\texttt{used\_once} : \mathcal{N} \to [\{\mathcal{N}\} \to \{0,1\}]$$

2.2 Initial Vectors

- $\mathcal{K}$: key space
- $\mathcal{N}$: nonce space
- $\mathcal{P}$: plaintext space
- $\mathcal{C}$: ciphertext space

Random IV Let $p \in \mathcal{P}$ and $k \in \mathcal{K}$.

$$\Pr\left[E_k(p, IV) = c\right] \approx \frac{1}{|\mathcal{C}|} \quad \text{for all } c \in \mathcal{C}.$$

Nonce IV Let $p, q \in \mathcal{P}$ and $n, m \in \mathcal{N}$.

$$\Pr\left[E_k(p, n) = c = E_k(q, m)\right] = 0 \quad \text{if } p \neq q, n \neq m.$$

$$\neg(a \vee b) \iff \neg a \wedge \neg b$$

`split.`	$\neg(a \vee b) \Rightarrow \neg a \wedge \neg b$
`move => not_or.`	$\neg a \wedge \neg b$
`split.`	$\neg a$
`case a.`	$a \Rightarrow \neg\top$
`move => a_true.`	$\neg\top$
$\vdots$	$\vdots$

1. $\neg(a \vee b) \Rightarrow \neg(a \wedge b)$
2. $\neg(a \vee b) \Rightarrow \neg(a \wedge b)$

Chapter 3

Installing EasyCrypt

[1]

EasyCrypt is a proof assistant for mechanizing proofs of the security of cryptographic constructions and protocols.

The official EasyCrypt installation instructions are available on the EasyCrypt GitHub. Below is a summary of these instructions that also emphasizes the connection with the Emacs text editor. EasyCrypt can be run from the shell (command line) in batch mode, to check individual .ec files. But when proofs are constructed interactively this is done within Emacs, with the generic interface Proof General mediating between Emacs and EasyCrypt, which is running as a sub-process of Emacs.

These instructions are current for:

- version 5.1.1 of the OCaml compiler;
- version 1.7.0 of why3;
- version 2.5.2 of alt-ergo.

(EasyCrypt is implemented in OCaml, why3 is the interface to SMT solvers used by EasyCrypt, and alt-ergo is one of SMT solvers you will need.)

Chapter 4

Ambient Logic

Note (Logics)**.** EasyCrypt has four logics:

- a **Probabilistic Relational Hoare Logic (pRHL)** for proving relations between pairs of procedures
- a **Probabilistic Hoare Logic (pHL)** for proving probabilistic facts about single procedures
- an **Ordinary Hoare Logic (HL)**
- an **Ambient Higher-order Logic** for proving mathematical facts and connecting judgements from the other logics
  - Based on higher order classical logic

Note (Proofs and Theories)**.**

- Proofs are structured as sequences of lemmas
- Lemmas are proved using tactics, as in Coq
  - Simple ambient logic goals can be proved using SMT solvers
- EasyCrypt theories may be used to group definitions, modules and lemmas together
- Theories may be specialized via cloning
  - Any axioms must then be proved

4.1 Types

EasyCrypt 's types include basic types like

- `unit` (which only has the single element `()`),
- `int`,
- `bool` and
- `real`,

as well as

- $\mathtt{t}_1 * \mathtt{t}_2 \cdots * \mathtt{t}_n$ and
- function types $\mathtt{t}_1 \rightarrow \mathtt{t}_2$.

'$*$' has higher precedence than $\rightarrow$, and $\rightarrow$ is right associative. Thus

$$\mathtt{t}_1 * \mathtt{t}_2 \rightarrow \mathtt{t}_3 \rightarrow \mathtt{t}_4$$

means $(\mathtt{t}_1 * \mathtt{t}_2) \rightarrow (\mathtt{t}_3 \rightarrow \mathtt{t}_4)$. A value of this type is a function that takes in a pair `(x,y)`, where `x` has type $\mathtt{t}_1$ and `y` has type $\mathtt{t}_2$, and returns a function that takes in a value `z` of type $\mathtt{t}_3$, and returns a result of type $\mathtt{t}_4$.

Appendix A

Boolean Functions

Bibliography

[1] Alley Stoughton. Easycrypt installation instructions, 2023. Accessed: 2024-07-12.