Automatic Proofs for Symmetric Encryption Modes

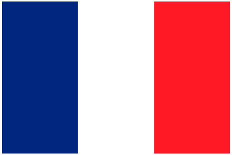
Martin Gagné[2] **Pascal Lafourcade**[1] Yassine Lakhnech[1]
Reihaneh Safavi-Naini[2]

[1] Université Grenoble 1, CNRS, VERIMAG, FRANCE

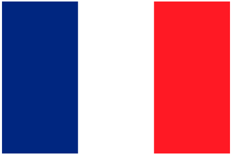
[2] Department of Computer Science, University of Calgary, Canada

3rd Canada-France Workshop on
Foundations & Practice of Security
June 22, 2010
Toronto.

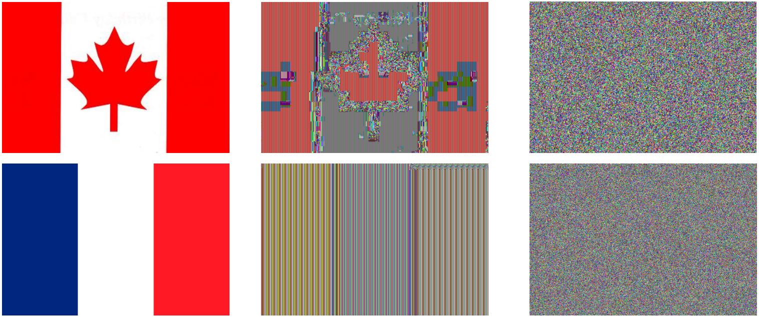
Indistinguishability and Symmetric Encryption Modes



Indistinguishability and Symmetric Encryption Modes



Indistinguishability and Symmetric Encryption Modes



ECB CBC, OFB ...

Block Cipher Modes

PRP $\mathcal{E} \rightarrow$ Encryption Mode $\rightarrow$ IND-CPA

NIST standard

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher FeedBack mode (CFB)
- Output FeedBack (OFB), and
- Counter mode (CTR).

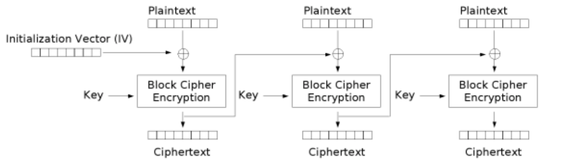
Others

DMC,CBC-MAC, IACBC, IAPM, XCB ,TMAC, HCTR, HCH, EME, EME*, PEP, OMAC, TET, CMC, GCM, EAX, XEX, TAE, TCH, TBC, CCM, ABL4

Block Cipher Modes

Example

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) mode encryption

$$C_i = \mathcal{E}(P_i \oplus C_{i-1}), C_0 = IV$$

CBC and others

CBC	CTR	OFB	CFB
$IV \xleftarrow{\$} \mathcal{U};$	$IV \xleftarrow{\$} \mathcal{U};$	$IV \xleftarrow{\$} \mathcal{U};$	$IV \xleftarrow{\$} \mathcal{U};$
$z_1 := IV \oplus m_1;$	$z_1 := \mathcal{E}(IV + 1);$	$z_1 := \mathcal{E}(IV);$	$z_1 := \mathcal{E}(IV);$
$c_1 := \mathcal{E}(z_1);$	$c_1 := m_1 \oplus z_1;$	$c_1 := m_1 \oplus z_1;$	$c_1 := m_1 \oplus z_1;$
$z_2 := c_1 \oplus m_2;$	$z_2 := \mathcal{E}(IV + 2);$	$z_2 := \mathcal{E}(z_1);$	$z_2 := \mathcal{E}(c_1);$
$c_2 := \mathcal{E}(z_2);$	$c_2 := m_2 \oplus z_2;$	$c_2 := m_2 \oplus z_2;$	$c_2 := m_2 \oplus z_2;$
$z_3 := c_2 \oplus m_3;$	$z_3 := \mathcal{E}(IV + 3);$	$z_3 := \mathcal{E}(z_2);$	$z_3 := \mathcal{E}(c_2);$
$c_3 := \mathcal{E}(z_3);$	$c_3 := m_3 \oplus z_3;$	$c_3 := m_3 \oplus z_3;$	$c_3 := m_3 \oplus z_3;$

Outline

Outline

How to prove an encryption mode is IND-CPA ?

Our Approach

Automated method for proving correctness of encryption mode:

- Language: Generic Encryption Mode
- Predicates: E, Indis, Lcounter
- Hoare logic : few rules

RESULT:

If a Generic Encryption Mode $\mathcal{E}_M$ is correct according to our Hoare logic then $\mathcal{E}_M$ is IND-CPA.

Grammar

$$
\begin{aligned}
\mathsf{c} \quad ::= \quad & x \xleftarrow{\$} \mathcal{U} \mid x := \mathcal{E}(y) \mid x := y \oplus z \mid x := y \| z \mid \\
& x := y + 1 \mid c_1; c_2
\end{aligned}
$$

Generic Encryption Mode

Definition

A generic encryption mode M is represented by

$$\mathcal{E}_M(m_1|\ldots|m_p, c_0|\ldots|c_p) : \textbf{var } \vec{x}; \text{c}$$

$\mathcal{E}_{CBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$:
var z_1, z_2, z_3;
$IV \xleftarrow{\$} \mathcal{U}$;
$z_1 := IV \oplus m_1$;
$c_1 := \mathcal{E}(z_1)$;
$z_2 := c_1 \oplus m_2$;
$c_2 := \mathcal{E}(z_2)$;
$z_3 := c_2 \oplus m_3$;
$c_3 := \mathcal{E}(z_3)$;

Predicates

$$\varphi ::= \mathsf{true} \mid \varphi \wedge \varphi \mid \psi$$
$$\psi ::= \mathsf{Indis}(\nu x; V) \mid \mathit{Seed}(e) \mid \mathsf{Lcounter}(x) \mid$$

Indis(νx; V): The value of x is indistinguishable from a random value given the value of the variables in V.

Seed(e): The probability that the value of e have been encrypted by $\mathcal{E}$ is negligible.

Lcounter(e): e is the most recent value of a monotone counter that started at a fresh random value.

Definition

Definition

Using previous notions we definie the two following predicates:

- $Useed(x) = Seed(x) \wedge \mathrm{Indis}(x)$
- $Cseed(x) = Seed(x) \wedge \mathrm{Lcounter}(x)$

Definition

Definition

Using previous notions we definie the two following predicates:

- $Useed(x) = Seed(x) \wedge \mathrm{Indis}(x)$
- $Cseed(x) = Seed(x) \wedge \mathrm{Lcounter}(x)$

Lemma

According to the defintions we have immediately:

- $Indis(\nu x) \Rightarrow Lcounter(x)$
- $Useed(x) \Rightarrow Cseed(x)$

More Formally

- $X \models \text{true}$.
- $X \models \varphi \wedge \varphi'$ iff $X \models \varphi$ and $X \models \varphi'$.
- $X \models \text{Indis}(\nu x; V)$ iff $[(S, \mathcal{E}) \xleftarrow{r} X : (S(x, V), \mathcal{E})] \sim [(S, \mathcal{E}) \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U}; S' = S\{x \mapsto u\} : (S'(x, V), \mathcal{E})]$
- $X \models \mathit{Seed}(x)$ iff $\Pr[(S, \mathcal{E}) \xleftarrow{r} X : S(x) \in S(\mathcal{T}_E).\mathit{dom}]$ is negligible.
- $X \models \text{Lcounter}(x)$ iff $\text{Indis}(x; \text{Var} \setminus \mathit{Tab}[x])$, where Tab[x] denote all variables that appear in table Tab[x] of $\mathcal{TF}$ until the variable x.

Semantics of the Programming Language

$$
\begin{array}{l}
[\![x \xleftarrow{r} \mathcal{U}]\!](S,\mathcal{E}) = [u \xleftarrow{r} \mathcal{U} : (S\{x \mapsto u, \mathcal{TF} \mapsto \mathcal{TF} \cup \{Tab[x]\}, \mathcal{E})] \\
[\![x := \mathcal{E}(y)]\!](S,\mathcal{E}) = \\
\qquad \begin{cases} \delta(S\{x \mapsto v, \mathcal{TF}, \mathcal{E}) \text{ if } (S(y),v) \in \mathcal{T}_E \\ \delta(S\{x \mapsto v, \mathcal{TF} \mapsto \mathcal{TF} \cup \{Tab[x]\}, \mathcal{T}_E \mapsto S(\mathcal{T}_E) \cdot (S(y),v)\}, \mathcal{E}) \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } (S(y),v) \notin \mathcal{T}_E \text{ and } v = \mathcal{E}(S(y)) \end{cases} \\
[\![x := y \oplus z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y) \oplus S(z), \mathcal{TF}, \mathcal{E}) \\
[\![x := y||z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y)||S(z), \mathcal{TF}, \mathcal{E}) \\
[\![x := y[n,m]]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y)[n,m], \mathcal{TF}, \mathcal{E}) \\
[\![x := y+1]\!](S,\mathcal{E}) = \\
\qquad \begin{cases} \delta(S\{x \mapsto S(y)+1, \mathcal{TF} \mapsto \mathcal{TF} \cup \{Tab[z] \mapsto Tab[z][i+1] = Tab[z][i+1] \cup x\}, \mathcal{E}) \\ \text{ if } y \in Tab[z][i] \\ \delta(S\{x \mapsto S(y)+1, \mathcal{TF}, \mathcal{E}) \text{ otherwise} \end{cases} \\
[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!]
\end{array}
$$

Table: The semantics of the programming language

How to generate $Seed(x)$?

Sampling a Random

(R1) $\{true\}\ x \stackrel{\$}{\leftarrow} \mathcal{U}\ \{Useed(x)\}$

How to generate *Seed*(x)?

Sampling a Random

(R1) $\{true\}\ x \stackrel{\$}{\leftarrow} \mathcal{U}\ \{Useed(x)\}$

PRP Encryption

(B1) $\{Seed(y)\}\ x := \mathcal{E}(y)\ \{Seed(x)\}$
(B2) $\{Seed(y)\}\ x := \mathcal{E}(y)\ \{\mathrm{Indis}(x)\}$

How to generate $Seed(x)$?

Xor

(X4) $\{\text{Indis}(x) \wedge Seed(x)\}\ z := x \oplus y\ \{Seed(z)\}$ if $y \neq z$
(X5) $\{\text{Lcounter}(t)\}\ z := x \oplus y\ \{\text{Lcounter}(t)\}$

How to generate $Seed(x)$?

Xor

(X4) $\{\mathrm{Indis}(x) \wedge Seed(x)\}\ z := x \oplus y\ \{Seed(z)\}$ if $y \neq z$
(X5) $\{\mathrm{Lcounter}(t)\}\ z := x \oplus y\ \{\mathrm{Lcounter}(t)\}$

Counter

- (I1) $\{\mathrm{Lcounter}(x)\}\ y := x + 1\ \{\mathrm{Lcounter}(y)\}$
- (I2) $\{lcounter(x)\}\ z := y + 1\ \{Seed(x)\}$

20 Rules

$x \xleftarrow{\$} \mathcal{U}$	$x = y\|\|z$	$x := y+1$		$x := y \oplus z$	$x := \mathcal{E}(y)$
(R1)	(C1)	(I1)	(G1)	(X1)	(B1)
(R2)	(C2)	(I2)	(G2)	(X2)	(B2)
		(I3)	(G3)	(X3)	(B3)
			(G4)	(X4)	(B4)
				(X5)	(B5)
					(B6)

Outline

How to prove that a Generic Encryption Mode is IND-CPA?

Theorem

Let $\mathcal{E}_M(m_1|\ldots|m_p, c_0|\ldots|c_p)$: **var** $\vec{x}$; c be a generic encryption mode, Then $\mathcal{E}_M$ is IND-CPA secure, if $\{\text{true}\}c \bigwedge_{i=0}^{i=p}\{\text{Indis}(\nu c_i; m_1, \ldots, m_p, c_0, \ldots, c_p)\}$ is valid.

Prototype

Implementation of a backward analysis in 1000 lines of Ocaml.

Examples

- CBC, FBC, OFB CFB are proved IND-CPA
- ECB and variants our tool fails: precondition is not true

All examples are immediate (less than one second)

Outline

Summary

- Generic Encryption Mode
- New predicats
- Hoare Logic for proving generic encryption mode IND-CPA
- Ocaml Prototype

Future Works

- Considering : For loops
- Hybrid encryption
- using Hash function
- using mathematics (GMC)
- IND-CCA ?
  Desai 2000: New Paradigms for Constructing Symmetric Encryption Schemes Secure against Chosen-Ciphertext Attack
- CBC-MAC

Thank you for your attention

Questions ?