Computer-aided cryptographic proofs

Gilles Barthe
IMDEA Software Institute, Madrid, Spain

March 24, 2014

The two views of cryptography

Computational cryptography

- Strong ties with complexity theory
- Feasible adversary breaks scheme with small probability
- Design of secure and efficient primitives and protocols
- Complex and manual proofs

The two views of cryptography

Computational cryptography

- Strong ties with complexity theory
- Feasible adversary breaks scheme with small probability
- Design of secure and efficient primitives and protocols
- Complex and manual proofs

Symbolic cryptography (Dolev and Yao, 1983)

- Assume perfect cryptography
- Adversary cannot win
- Discovery of logical bugs in protocols
- Strong ties with verification
- Automated proofs

Reconciling the two views

(Abadi and Rogaway, 2000)

Computational soundness

Security in symbolic model implies computational security

- ...under non-standard assumptions on primitives
- Symbolic tools deliver asymptotic guarantees
- ...but no concrete guarantees
- Applicable to many settings
- ...but some impossibility results

Computational proof systems

Prove security

- directly in the computational model
- under standard assumptions

- Indistinguishability logics
- CryptoVerif (game-playing approach for applied π-calculus)
- EasyCrypt (code-based game-playing approach)
- Domain-specific logics

Why bother?

- *In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor*. Bellare and Rogaway, 2004-2006
- *Do we have a problem with cryptographic proofs? Yes, we do [...] We generate more proofs than we carefully verify (and as a consequence some of our published proofs are incorrect)*. Halevi, 2005

Optimal Asymmetric Encryption Padding

Encryption $\mathcal{E}_{\mathrm{OAEP}(pk)}(m)$:

$r \xleftarrow{\$} \{0,1\}^{k_0}$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
return $\mathsf{f}_{pk}(s \| t)$

Decryption $\mathcal{D}_{\mathrm{OAEP}(sk)}(c)$:

$(s,t) \leftarrow \mathsf{f}_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $([s \oplus G(r)]_{k_1} = 0^{k_1})$
  then $\{m \leftarrow [s \oplus G(r)]^{k}; \}$
  else $\{m \leftarrow \perp; \}$
return m

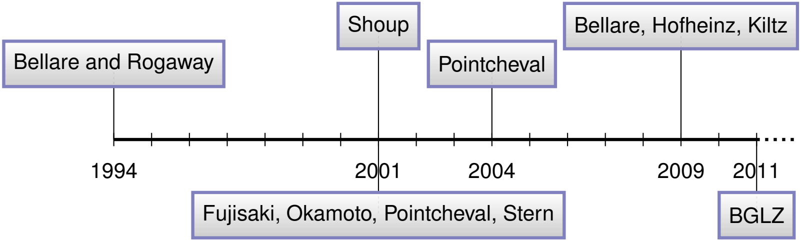
FOR ALL IND-CCA adversary $\mathcal{A}$ against $(\mathcal{K}, \mathcal{E}_{\mathrm{OAEP}}, \mathcal{D}_{\mathrm{OAEP}})$,
THERE EXISTS a sPDOW adversary $\mathcal{I}$ against $(\mathcal{K}, \mathsf{f}, \mathsf{f}^{-1})$ st

$$\left|\Pr_{\text{IND-CCA}(\mathcal{A})}[b' = b] - \frac{1}{2}\right| \leq$$
$$\Pr_{\text{PDOW}(\mathcal{I})}[y \in Y'] + \frac{3q_D q_G + q_D^2 + 4q_D + q_G}{2^{k_0}} + \frac{2q_D}{2^{k_1}}$$

and

$$t_{\mathcal{I}} \leq t_{\mathcal{A}} + q_D\, q_G\, q_H\, T_{\mathsf{f}}$$

OAEP: provable security



1994 Purported proof of chosen-ciphertext security

2001 1994 proof gives weaker security; desired security holds

- for a modified scheme
- under stronger assumptions

2004 Filled gaps in 2001 proof

2009 Security definition needs to be clarified

2011 Fills gaps in 2004 proof

Computer-aided cryptographic proofs

provable security
=
deductive verification of parametrized probabilistic programs

- adhere to cryptographic practice
  - ☞ same proof techniques
  - ☞ same guarantees
  - ☞ same level of abstraction
- leverage existing verification techniques and tools
  - ☞ program logics, VC generation, invariant generation
  - ☞ SMT solvers, theorem provers, proof assistants
  - ☞ symbolic cryptography

A language for cryptographic games

$\mathcal{C}$	$::=$	skip	skip
	$\mid$	$\mathcal{V} \leftarrow \mathcal{E}$	assignment
	$\mid$	$\mathcal{V} \stackrel{\$}{\leftarrow} \mathcal{D}$	random sampling
	$\mid$	$\mathcal{C};\ \mathcal{C}$	sequence
	$\mid$	if $\mathcal{E}$ then $\mathcal{C}$ else $\mathcal{C}$	conditional
	$\mid$	while $\mathcal{E}$ do $\mathcal{C}$	while loop
	$\mid$	$\mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E})$	procedure call

- $\mathcal{E}$: (higher-order) expressions
- $\mathcal{D}$: discrete sub-distributions

user extensible

- $\mathcal{P}$: procedures
  - oracles: concrete procedures
  - adversaries: constrained abstract procedures

Reasoning about programs

- Probabilistic Hoare Logic

$$\vDash \{P\} c \{Q\} \diamond \delta$$

- Probabilistic Relational Hoare logic

$$\vDash \{P\}\ c_1\ \sim\ c_2\ \{Q\}$$

- Ambient logic

Selected rules

$$\frac{\models \{P\}\ c_1\ \sim\ c_2\ \{Q\} \qquad \models \{Q\}\ c_1'\ \sim\ c_2'\ \{R\}}{\models \{P\}\ c_1; c_1'\ \sim\ c_2; c_2'\ \{R\}}$$

$$\frac{\models \{P \wedge e\langle 1\rangle\}\ c_1\ \sim\ c\ \{Q\} \qquad \models \{P \wedge \neg e\langle 1\rangle\}\ c_2\ \sim\ c\ \{Q\}}{\models \{P\}\ \text{if } e \text{ then } c_1 \text{ else } c_2\ \sim\ c\ \{Q\}}$$

$$\frac{\begin{array}{c} P \rightarrow e\langle 1\rangle = e'\langle 2\rangle \\ \models \{P \wedge e\langle 1\rangle\}\ c_1\ \sim\ c_1'\ \{Q\} \quad \models \{P \wedge \neg e\langle 1\rangle\}\ c_2\ \sim\ c_2'\ \{Q\} \end{array}}{\models \{P\}\ \text{if } e \text{ then } c_1 \text{ else } c_2\ \sim\ \text{if } e' \text{ then } c_1' \text{ else } c_2'\ \{Q\}}$$

$$\frac{f \text{ is 1-1}}{\models \{\forall v, Q[x\langle 1\rangle := f\ v, x\langle 2\rangle := v]\}\ x \xleftarrow{\$} A\ \sim\ x \xleftarrow{\$} A\ \{Q\}}$$

Applications

Allows deriving judgments of the form

$$\Pr_{c_1,m_1}[A_1] \leq \Pr_{c_2,m_2}[A_2]$$

or

$$|\Pr_{c_1,m_1}[A_1] - \Pr_{c_2,m_2}[A_2]| \leq \Pr_{c_2,m_2}[F]$$

Benefits

- Application of rules directed by syntax (mostly)
- Can be automated
- Generate verification conditions
- VCs can be discharged by SMT solvers

Building on 40 years of research in program verification!

Example: Bellare and Rogaway 1993 encryption

Game IND-CPA($\mathcal{A}$) :
$(sk, pk) \leftarrow \mathcal{K}(\ );$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$s \leftarrow H(r) \oplus m;$
$y \leftarrow f_{pk}(r) \,\|\, s;$
return y

For every IND-CPA adversary $\mathcal{A}$, there exists an inverter $\mathcal{I}$ st

$$\Pr_{\text{IND-CPA}(\mathcal{A})}\left[b' = b\right] - \frac{1}{2} \leq \Pr_{\text{OW}(\mathcal{I})}\left[y' = y\right]$$

Proof

Game hopping technique

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \stackrel{\$}{\leftarrow} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \stackrel{\$}{\leftarrow} \{0,1\}^\ell;$
$h \leftarrow H(r);$
$s \leftarrow h \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game G :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \stackrel{\$}{\leftarrow} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \stackrel{\$}{\leftarrow} \{0,1\}^\ell;$
$h \stackrel{\$}{\leftarrow} \{0,1\}^k;$
$s \leftarrow h \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game G$'$:
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \stackrel{\$}{\leftarrow} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \stackrel{\$}{\leftarrow} \{0,1\}^\ell;$
$s \stackrel{\$}{\leftarrow} \{0,1\}^k;$
$h \leftarrow s \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}();$
$y \stackrel{\$}{\leftarrow} \{0,1\}^\ell;$
$y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$s \stackrel{\$}{\leftarrow} \{0,1\}^k;$
$c^\star \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(c^\star);$
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x];$
return y'

1. For each hop
   - prove validity of pRHL judgment
   - derive probability claims
   - (possibly) resolve some probability expressions using pHL
2. Obtain security bound by combining claims
3. Check execution time of constructed adversary

Conditional equivalence

$$\begin{array}{l} \mathcal{E}_{pk}(m): \\ \quad r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell}; \\ \quad h \leftarrow H(r); \\ \quad s \leftarrow h \oplus m; \\ \quad c \leftarrow f_{pk}(r) \,\|\, s; \\ \quad \text{return } c \end{array} \longrightarrow \begin{array}{l} \mathcal{E}_{pk}(m): \\ \quad r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell}; \\ \quad h \stackrel{\$}{\leftarrow} \{0,1\}^{k}; \\ \quad s \leftarrow h \oplus m; \\ \quad c \leftarrow f_{pk}(r) \,\|\, s; \\ \quad \text{return } c \end{array}$$

$$\models \{\top\}\ \text{IND-CPA} \sim \mathbf{G}\ \left\{(\neg r \in L_H^{\mathcal{A}})\langle 2 \rangle \ \rightarrow =_{b,b'}\right\}$$

$$\Pr_{\text{IND-CPA}}\left[b' = b\right] - \Pr_{\mathbf{G}}\left[b' = b\right] \leq \Pr_{\mathbf{G}}\left[r \in L_H^{\mathcal{A}}\right]$$

Equivalence

$$\begin{array}{l}\mathcal{E}_{pk}(m):\\ \quad r \xleftarrow{\$} \{0,1\}^{\ell};\\ \quad h \xleftarrow{\$} \{0,1\}^{k};\\ \quad s \leftarrow h \oplus m;\\ \quad c \leftarrow f_{pk}(r) \parallel s;\\ \quad \text{return } c\end{array} \longrightarrow \begin{array}{l}\mathcal{E}_{pk}(m):\\ \quad r \xleftarrow{\$} \{0,1\}^{\ell};\\ \quad s \xleftarrow{\$} \{0,1\}^{k};\\ \quad h \leftarrow s \oplus m;\\ \quad c \leftarrow f_{pk}(r) \parallel s;\\ \quad \text{return } c\end{array}$$

$$\models \{\top\}\ \mathbf{G} \sim \mathbf{G}'\ \left\{=_{b,b',r,L_H^{\mathcal{A}}}\right\}$$

$$\Pr_{\mathbf{G}}\left[r \in L_H^{\mathcal{A}}\right] = \Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right] \qquad \Pr_{\mathbf{G}}[b' = b] = \Pr_{\mathbf{G}'}[b' = b] = \frac{1}{2}$$

Equivalence

$$
\begin{array}{l}
\mathcal{E}_{pk}(m):\\
\quad r \xleftarrow{\$} \{0,1\}^{\ell};\\
\quad h \xleftarrow{\$} \{0,1\}^{k};\\
\quad s \leftarrow h \oplus m;\\
\quad c \leftarrow f_{pk}(r) \,\|\, s;\\
\quad \text{return } c
\end{array}
\quad\Longrightarrow\quad
\begin{array}{l}
\mathcal{E}_{pk}(m):\\
\quad r \xleftarrow{\$} \{0,1\}^{\ell};\\
\quad s \xleftarrow{\$} \{0,1\}^{k};\\
\quad h \leftarrow s \oplus m;\\
\quad c \leftarrow f_{pk}(r) \,\|\, s;\\
\quad \text{return } c
\end{array}
$$

$$\models \{\top\}\ \mathbf{G} \sim \mathbf{G}' \left\{=_{b,b',r,L_H^{\mathcal{A}}}\right\}$$

$$\Pr_{\text{IND-CPA}}[b' = b] - \frac{1}{2} \leq \Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right]$$

Reduction

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$s \xleftarrow{\$} \{0,1\}^k;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^\ell;$
$y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$s \xleftarrow{\$} \{0,1\}^k;$
$c^\star \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(c^\star);$
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x];$
return y'

$$\models \{\top\}\ \mathbf{G}' \sim \mathrm{OW}\ \left\{(r \in L_H^{\mathcal{A}})\langle 1\rangle \rightarrow (y' = y)\langle 2\rangle\right\}$$

$$\Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right] \leq \Pr_{\mathrm{OW}(\mathcal{I})}[y' = y]$$

Reduction

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$s \xleftarrow{\$} \{0,1\}^k;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^\ell;$
$y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$s \xleftarrow{\$} \{0,1\}^k;$
$c^\star \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(c^\star);$
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x];$
return y'

$$\models \{\top\}\ \mathbf{G}' \sim \mathrm{OW}\ \left\{(r \in L_H^{\mathcal{A}})\langle 1\rangle \rightarrow (y' = y)\langle 2\rangle\right\}$$

$$\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \frac{1}{2} \leq \Pr_{\text{OW}(\mathcal{I})}[y' = y]$$

Case studies

- Public-key encryption
- Signatures
- Hash designs
- Block ciphers
- Zero-knowledge protocols
- Differential privacy

Based on

- CertiCrypt (2006-2011): Coq based
- EasyCrypt (2009-): SMT based

Limitations

- Abstraction and composition
- Implementation
- Optimization

Abstraction and composition

The need

pRHL can capture *instances of* generic proof steps

- Repeated proof effort; requires ability to reason in pRHL
- Does not scale to complex proofs
- Gap between formal proofs and cryptographic practice

Two mechanisms

- Module system
  - ☞ ML-like module system, with negative constraints
  - ☞ Allows hybrid arguments
  - ☞ Quantification over modules
- Type classes

Modules at work

```
module type Scheme = {
  fun kg () : skey * pkey
  fun enc(pk:pkey, m:plaintext) : ciphertext
  fun dec(sk:skey, c:ciphertext) : plaintext
}.

module type ADV = {
  fun choose (pk:pkey) : msg * msg
  fun guess (c:cipher) : bool
}.
```

```
module CPA (S:Scheme, A:ADV) = {
  fun main () : bool = {
    var pk,sk,m0,m1,b,b',challenge;
    (pk,sk) = S.kg();
    (m0,m1) = A.choose(pk);
    b <$- {0,1};
    challenge = S.enc(pk, b?m1:m0);
    b' = A.guess(challenge);
    return b' = b;
  }
}.
```

Reduction argument:

$\forall$ (A <: Adv), $\exists$ (I <: Inverter), Pr[CPA(A) : res] $-$ 1/2 $\leq$Pr[OW(I) : res].

Existential quantification over modules would require support for complexity claims

Implementation

Painful gap between provable security and real world

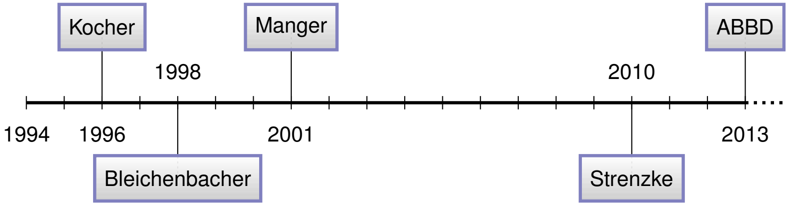
- Proofs reason about algorithmic descriptions
- Standards constrain implementations
- Attackers target executable code

Real-world crypto is breakable; is in fact being broken; is one of many ongoing disaster areas in security. Bernstein, 2013

Reasoning about implementations

- C-mode
- Use CompCert as a backend
- Account for side-channel countermeasures

OAEP: real-world security



- RSA implementations are vulnerable to timing attacks
- PKCS1.5 vulnerable to padding oracle attacks
- RSA is a permutation only on strict subset of $[0..2^k]$: careless conversion leads to timing attacks

Implementation of OAEP

Decryption $\mathcal{D}_{\mathrm{OAEP}(sk)}(c)$:

$(s, t) \leftarrow \mathsf{f}_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $([s \oplus G(r)]_{k_1} = 0^{k_1})$
  then $\{m \leftarrow [s \oplus G(r)]^{k};\}$
  else $\{m \leftarrow \perp;\}$
return m

Decryption $\mathcal{D}_{\mathrm{PKCS\text{-}C}(sk)}(res, c)$:

if ($c \in$ MsgSpace(sk)) then
$\{\ (b0, s, t) \leftarrow \mathsf{f}_{sk}^{-1}(c)$;
  $h \leftarrow MGF(s, hL)$; $i \leftarrow 0$;
  while $(i < hLen + 1)$
  $\{\ s[i] \leftarrow t[i] \oplus h[i];\ i \leftarrow i + 1;\ \}$
  $g \leftarrow MGF(r, dbL)$; $i \leftarrow 0$;
  while $(i < dbLen)$
  $\{\ p[i] \leftarrow s[i] \oplus g[i];\ \ i \leftarrow i + 1;\ \}$
  $l \leftarrow payload\_length(p)$;
  if $(b0 = 0^8 \wedge [p]_l^{hLen} = 0..01 \wedge$
      $[p]_{hLen} = LHash)$
    then
      $\{rc \leftarrow Success$;
      $memcpy(res, 0, p, dbLen - l, l);\}$
    else $\{rc \leftarrow DecryptionError;\}\ \}$
  else $\{rc \leftarrow CiphertextTooLong;\}$
return rc;

Provable security of C and executable code

C-mode for EasyCrypt

- Prove PKCS using C-mode for EasyCrypt: arrays are base-offset representation and match subset of C arrays (no aliasing or overlap possible, pointer arithmetic only within an array)
- Carry proof from C-like code to x86 executable

Reduction proof

FOR ALL adversary that breaks the executable code,
THERE EXISTS an adversary that breaks the source code

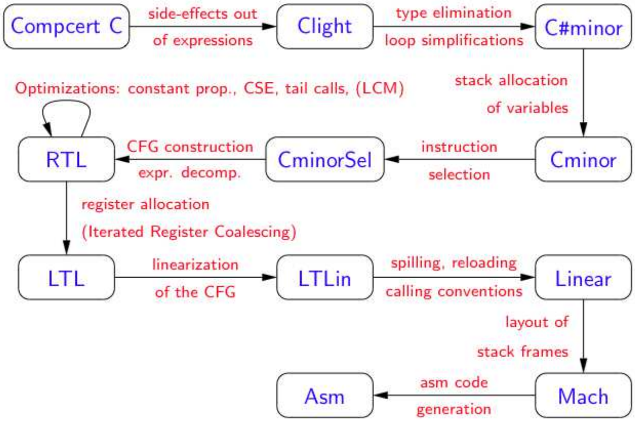
- Account for some side-channels

Certified compilers

Reduction proof rests on semantic preservation. Use CompCert

CompCert (Leroy, 2006)

- Optimizing C compiler implemented in Coq
- Formal proof of semantic preservation



Application to OAEP

Baseline security

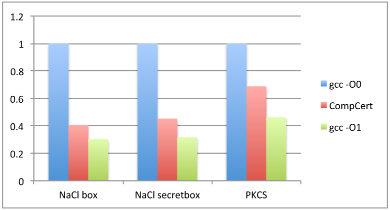
- idealized operations moved to the environment
  - ☞ random sampling of bitstrings
  - ☞ hash function (random oracle)
- "trusted-lib" mechanism for arithmetic libraries

PC security

- annotation mechanism is used to track control flow
- ASM analyser checks that compiler does not add branches
- "trusted-lib" must also verify leakage assumptions

Implementations and performance

- Carefully craft C code to avoid hidden branching (operations on bitstrings or booleans)
- Compile with CompCert
- Check no new branch was introduced
- Links generated code with LIP (secure in PC model)



Cache-based side-channel attacks

A recipe for security disaster

- Array accesses with high indices
- Adversary gains knowledge through cache
- Lead to devastating cache-based attacks:
  AES, BlowFish, DES, RC4, Snow…
- Alternative: constant-time crypto

Protection mechanism

- alias and information flow analysis (CompCert assembly)
- tag arrays accessed with secret indices as "stealth"
- typable programs w/o "stealth" address are constant-time
- typable programs execute securely on stealth memory
  (provided stealth addresses are mapped correctly)

Examples

Example	LoC	# stealth addresses	Size (KB)
AES	744	5	4
DES	836	10	2
Blowfish	279	1	4
RC4	164	1	0.25
Snow	757	6	6
Salsa20	1077	0	0
TEA	70	0	0
SHA256	419	0	0

Automated synthesis of cryptosystems

Do the cryptosystems reflect [...] the situations that are being catered for? Or are they accidents of history and personal background that may be obscuring fruitful developments? [...] We must systematize their design so that a new cryptosystem is a point chosen from a well-mapped space, rather than a laboriously devised construction. (Adapted from Landin, 1966. The next 700 programming languages)

An algebraic view of padding-based schemes

Encryption algorithms are modelled as algebraic expressions

$$
\begin{array}{rcll}
\mathcal{E} & ::= & m & \text{input message} \\
& | & 0 & \text{zero bitstring} \\
& | & \mathcal{R} & \text{uniform random bitstring} \\
& | & \mathcal{E} \oplus \mathcal{E} & \text{xor} \\
& | & \mathcal{E} \parallel \mathcal{E} & \text{concatenation} \\
& | & [\mathcal{E}]_s^s & \text{projection} \\
& | & H(\mathcal{E}) & \text{hash} \\
& | & f(\mathcal{E}) & \text{trapdoor permutation}
\end{array}
$$

Decryption algorithms are modelled in a mild extension

Semantics

Left-to-right evaluation with sharing, yields a pWHILE procedure

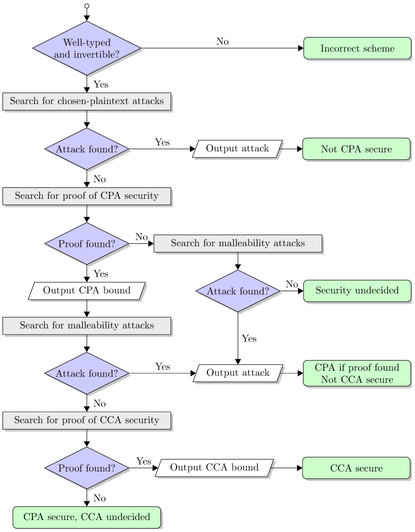
Example

$$f((G(r) \oplus (m \| 0)) \| H(G(r) \oplus (m \| 0)) \oplus r)$$

interpreted as:

$$\begin{array}{l} r \stackrel{\$}{\leftarrow} \{0,1\}^k; \\ g \leftarrow G(r); \\ s \leftarrow g \oplus (m \| 0); \\ h \leftarrow H(s); \\ \text{return } f_{pk}(s \| (h \oplus r)) \end{array}$$

Approach


Well-typed and invertible?
No
Incorrect scheme
Yes
Search for chosen-plaintext attacks
Attack found?
Yes
Output attack
Not CPA secure
No
Search for proof of CPA security
Proof found?
No
Search for malleability attacks
Attack found?
No
Security undecided
Yes
Output CPA bound
Search for malleability attacks
Attack found?
Yes
Output attack
CPA if proof found
Not CCA secure
No
Search for proof of CCA security
Proof found?
Yes
Output CCA bound
CCA secure
No
CPA secure, CCA undecided

Attack finding

- Based on standard symbolic tools
  - ☞ deducibility and static equivalence
- For efficiency, first apply simple filters, eg
  - ☞ is decryption possible without a key? $m \parallel f(r)$
  - ☞ is encryption randomized? $f(m)$
  - ☞ is randomness extractable without a key? $r \parallel f(m \oplus r)$

$$\frac{e \vdash e_1 \quad e \vdash e_2}{e \vdash e_1 \parallel e_2}[\text{Conc}] \qquad \frac{e \vdash e_1 \qquad e \vdash e_2}{e \vdash e_1 \oplus e_2}[\text{Xor}]$$

$$\frac{e \vdash e}{e \vdash [e]_n^\ell}[\text{Proj}] \qquad \frac{e \vdash e_1 \qquad \vdash e_1 \doteq e_2}{e \vdash e_2}[\text{Conv}]$$

$$\frac{e \vdash e'}{e \vdash H(e')}[\text{H}] \qquad \frac{e \vdash e'}{e \vdash f(e')}[\text{F}] \qquad \boxed{\frac{e \vdash e'}{e \vdash f^{-1}(e')}[\text{Finv}]}$$

Chosen-plaintext security: principles

Failure event	Replace $H(e)$ by fresh r
Optimistic sampling	Replace $e \oplus r$, where r is fresh, by r
Probability	Compute probability of $b = b'$ or $e \in L$
Reduction	Find inverter and apply one-wayness

Chosen-plaintext security: formalism

Judgment

$$c :_p \varphi$$

- Reasons about probability of events
- Concrete probability can be computed
  - ☞ on the fly
  - ☞ a posteriori (judgments use only $p = 0, \frac{1}{2}$)
- Side-conditions are discharged by symbolic methods
  - ☞ what is the entropy of e?
  - ☞ can I compute e' from e?

Chosen-plaintext security: proof rules

$$\frac{m \notin c^\star}{c^\star :_{\frac{1}{2}} \text{Guess}}\text{[Indep]}$$

$$\frac{e \vdash \vec{r} \qquad \vec{r} \cap \mathcal{R}(c^\star) = \emptyset}{c^\star :_0 e \in L_H^{\mathcal{A}}}\text{[Indom]}$$

$$\frac{c^\star :_p \varphi \quad r \text{ fresh}}{(c^\star :_p \varphi)\{e \oplus r / r\}}\text{[Opt]}$$

$$\frac{e \vdash^{\mathcal{A}} \vec{r} \qquad f(\vec{r}) \parallel \vec{r_0} \parallel m \vdash^{\mathcal{A}} c^\star \qquad \vec{r} \cap \vec{r_0} = \emptyset}{c^\star :_0 e \in L_H^{\mathcal{A}}}\text{[OW]}$$

Evaluation

	Proof	Attack	Unknown
CPA	116433	901929	12627
CCA	28001	182730	66332

Remarks:

- 16185 out of 66332 CCA unknown non-redundant
- CPA unknown seem secure
- Proving completeness seems very hard
- Found (by inspection) one new and interesting scheme

$$f(r \parallel m \oplus G(r))$$

- Missing methods for mining the database

Some ongoing projects

- Foundations and automation
- Case studies: AKE, MPC, faults, verifiable computation
- Synthesis of dlog and pairing-based encryption
- Verified batch verifiers
- Automated analysis in (multi-linear) generic groups

Conclusion

- Solid foundation for cryptographic proofs
- Formal verification of emblematic case studies
- Narrowing the gap between proofs and code
- Automated analysis and synthesis of crypto schemes

`http://www.easycrypt.info`

EasyCrypt toolchain