Computer-aided cryptography

Gilles Barthe
IMDEA Software Institute, Madrid, Spain

May 1, 2017

Call for Papers
CRYPTO 2011

General Information

Original papers on all technical aspects of cryptology are solicited for submission to CRYPTO 2011, the 31st Annual International Cryptology Conference. Besides the usual topics, submissions are also welcome on topics not routinely appearing at recent CRYPTOs, including cryptographic work in the style of the CHES workshop or CSF symposium. CRYPTO 2011 is sponsored by the International Association for Cryptologic Research (IACR), in cooperation with the Computer Science Department of the University of California, Santa Barbara.

Computer-Aided Security Proofs for the Working Cryptographer*

Gilles Barthe[1], Benjamin Grégoire[2], Sylvain Heraud[2], and Santiago Zanella Béguelin[1]

- S. Halevi: A plausible approach to computer-aided cryptographic proofs
- M. Bellare and P. Rogaway: Code-Based Game-Playing Proofs and the Security of Triple Encryption
- V. Shoup: Sequences of Games: A Tool for Taming Complexity in Security Proofs

Computer-aided cryptography

Develop tool-assisted methodologies for helping the design, analysis, and implementation of cryptographic constructions (primitives and protocols)

Goals:

- Automated analysis of (symbolic or computational) security
- Independently verifiable proofs of (computational) security
- Verified implementations
- New designs and better implementations
- etc

Building on formal methods

- program analysis (safety)
- program verification (correctness)
- compilation (optimization)
- program synthesis
- etc

Potential benefits

Formal methods for cryptography

- higher assurance
- smaller gap between provable security and crypto engineering
- new proof techniques

Cryptography for formal methods

- Challenging and non-standard examples
- New theories and applications

A long-term goal

- FOR EVERY adversary that breaks assembly code,
- IF assembly code is safe and leakage resistent,
- AND assembly code correctly implements algorithm,
- THERE EXISTS an adversary that breaks the algorithm

Challenges:

- Models: execution, leakage, adversaries
- Practical: build efficient libraries
- Formal methods: theories and engineering

Current landscape

- Security in symbolic and computational model: ProVerif, Tamarin, CryptoVerif, EasyCrypt, F*...
- Side-channel analysis: ct-grind, ct-verif, FlowTracker, CacheAudit, Sleuth, maskcomp, maskverif
- Safety: TIS analyzer...
- Functional correctness: Cryptol, CompCert/VST, gf-verif...
- Cryptographic engineering: qhasm, boringssl, Charm...

Case study: MEE-CBC

- Black-box IND$-CPA security proof
- Equivalence w/ C implementation and specification
- Compile C using CompCert
- Apply certified constant-time verifier

Other examples: PKCS, HMAC, HACL*, miTLS

EasyCrypt

Domain-specific proof assistant

- proof goals tailored to reductionist proofs
- proof tools support common proof techniques (bridging steps, failure events, hybrid arguments, eager sampling. . .)

Control and automation from state-of-art verification

- interactive proof engine and mathematical libraries (a la Coq/ssreflect)
- back-end to SMT solvers and CAS

Game playing as (implicit) probabilistic couplings

Let $\mu_1, \mu_2 \in \mathsf{Dist}(A)$ and $R \subseteq A \times A$. Let $\mu \in \mathsf{Dist}(A \times A)$.

- μ is a coupling for (μ_1, μ_2) iff $\pi_1(\mu) = \mu_1$ and $\pi_2(\mu) = \mu_2$
- μ is a R-coupling for (μ_1, μ_2) if moreover $\Pr_{y \leftarrow \mu}[y \notin R] = 0$

Let μ is a R-coupling for (μ_1, μ_2).

- Bridging step: if R is equality, then for every event X,

$$\Pr_{z \leftarrow \mu_1}[X] = \Pr_{z \leftarrow \mu_2}[X]$$

- Failure Event: If $x \; R \; y$ iff $F(x) \Rightarrow x = y$ and $F(x) \Leftrightarrow F(y)$, then for every event X,

$$\left|\Pr_{z \leftarrow \mu_1}[X] - \Pr_{z \leftarrow \mu_2}[X]\right| \leq \max(\Pr_{z \leftarrow \mu_1}[\neg F], \Pr_{z \leftarrow \mu_2}[\neg F])$$

- Reduction: If $x \; R \; y$ iff $F(x) \Rightarrow G(y)$, then

$$\Pr_{x \leftarrow \mu_2}[G] \leq \Pr_{y \leftarrow \mu_1}[F]$$

Cryptographic proofs as probabilistic couplings

A useful insight?

- Prior (but limited) use of probabilistic couplings in crypto
- Key to build scalable verification infrastructure
  - No need to reason directly about probabilities
  - Make crypto proofs look "almost" like standard verification
- Helps generalizations (differential privacy, quantum crypto)

Code-based approach to probabilistic couplings

- Code-based approach

$$
\begin{array}{rcll}
\mathscr{C} & ::= & \mathsf{skip} & \text{skip} \\
& | & \mathscr{V} \leftarrow \mathscr{E} & \text{assignment} \\
& | & \mathscr{V} \xleftarrow{\$} \mathscr{D} & \text{random sampling} \\
& | & \mathscr{C};\ \mathscr{C} & \text{sequence} \\
& | & \mathsf{if}\ \mathscr{E}\ \mathsf{then}\ \mathscr{C}\ \mathsf{else}\ \mathscr{C} & \text{conditional} \\
& | & \mathsf{while}\ \mathscr{E}\ \mathsf{do}\ \mathscr{C} & \text{while loop} \\
& | & \mathscr{V} \leftarrow \mathscr{P}(\mathscr{E},\ldots,\mathscr{E}) & \text{procedure (oracle/adv) call}
\end{array}
$$

- Game-playing technique: $\models \{P\}\ c_1\ \sim\ c_2\ \{Q\}$ where P and Q are relations on states
- Concrete security: $\{\Psi\}c\{\Pr[\Phi] \leq \beta\}$ (many limitations)
- Bound execution time of constructed adversary (limited tool support)

Some proof rules

Conditionals

$$\frac{\models \{\Phi \wedge b_1 \wedge b_2\}\ c_1 \ \sim\ c_2\ \{\Psi\} \qquad \models \{\Phi \wedge \neg b_1 \wedge \neg b_2\}\ c_1' \ \sim\ c_2'\ \{\Psi\}}{\models \{\Phi \wedge b_1 = b_2\}\ \text{if } b_1 \text{ then } c_1 \text{ else } c_1' \ \sim\ \text{if } b_2 \text{ then } c_2 \text{ else } c_2'\ \{\Psi\}}$$

Random assignment

$$\frac{f \in T \xrightarrow{1-1} T \qquad \forall v \in T.\ \mu_1(v) = \mu_2(f\ v)}{\models \left\{\forall v, Q[v/x_1, f\ v/x_2]\right\}\ x_1 \xleftarrow{\$} \mu_1 \ \sim\ x_2 \xleftarrow{\$} \mu_2\ \{Q\}}$$

- Bijection f: specifies how to coordinate the samples
- Side condition: marginals are preserved under f

Status

- Broadly applicable: encryption, signatures, hash designs, key exchange protocols, zero-knowledge protocols, garbled circuits, SHA3, voting
- Helped unveiled subtle points in proofs
- Interactive tools remain time-consuming and difficult to use

A lightweight approach

- Probabilistic experiments
- Probabilistic inequalities
- Proofs

Formalization brings significant benefits at each stage

- Abstraction and automation (problem specific)

Highly automated proofs

Many high-level principles are guess-and-check:

- Bridging steps: guess couplings, check equivalence
- Reduction steps: guess adversary, check equivalence

Automation:

- Proof-producing equivalence checker
- Heuristics for guessing

AutoG&P

- Automated proofs for DDH-based cryptography
- Cramer-Shoup, Boneh-Boyen, structure-preserving encryption

Challenge

- Build sufficiently rich set of high-level rules
- Decision procedures
  (Jutla and Roy 2012, Carmer and Rosulek 2016)

Automated proofs in ROM

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

- Hard to get security proofs right
- 6 months to formalize the proof!
- Many variants in the literature
- About 200 variants of SAEP/OAEP (Komano and Ohta)
- About $10^6 - 10^8$ candidates schemes of "reasonable" size
- Can we automate analysis for finding attacks or proofs?

ZooCrypt

- Extremely efficient logics for CPA and CCA security (up-to-bad, optimistic sampling, reduction, reject some ciphertexts)
- Extremely efficient procedures for detecting attacks
- Smart generation of candidate constructions

Experiments

- Generated 1,000,000 candidates
- For CPA security: 99,5% solved by the tool
- For CCA security: 80% solved by tool
- Practical interpretation (sql database)
- Manual inspection for grey zone
- Interactive tutor

ZAEP

- OAEP (1994):

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

- SAEP (2001):

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

- ZAEP (2012):

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

  ☞ redundancy-free
  ☞ INDCCA secure for RSA with exponent 2 and 3

Automated proofs in GGM

- Introduced for proving lower bounds of DL algorithms
- Algorithms do not have direct access to algebraic values
- Used for validating hardness assumptions and efficient schemes
- Master theorem: symbolic security implies generic security
- Symbolic security by constraint solving (big operators)
- Applications: synthesis of SPS and ABE compiler

Timing attacks

- AES (Osvik, Shamir, Tromer 2006)
- MEE-CBC (AlFardan, Paterson 2013)
- RSA (Yarom, Falkner, 2014)
- ...

Work remotely!

Cryptographic constant-time

Control flow and memory accesses should be independent of secrets

However, cryptographic constant-time is hard to program

Case study: MEE-CBC s2n implementation

- number of calls to compression function during decryption must not depend on padding length or validity (Lucky 13)
- s2n performs some mitigation and adds random delay
- Insufficient in practice (Luckyμs). More mitigation
- Off-by-one error still causes large timing discrepancies, and leads to plaintext recovery

ct-verif

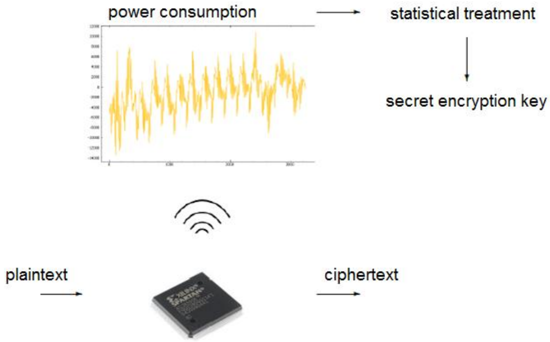
Product program

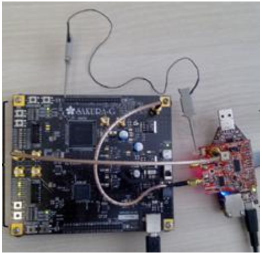
- Two copies of program in lockstep
- Check agreement at critical instructions (branching/memory)

Inspired from Zaks and Pnueli (2008)

- Sound and relatively complete
- Supports private and public outputs
- Implementation for LLVM, based on Smack
- Extensively evaluated: NaCl, OpenSSL, FourQ, SUPERCOP
- Ongoing: vector instructions, counter-example generation

Differential power analysis



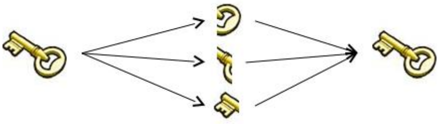


- Measure power consumption during execution
- Analysis of power can be used to recover secrets

Security models and masked implementations

- Threshold probing model: adversary can observe t-tuples of intermediate values
- Noisy leakage model: all instructions leak. Leakage is noisy

Models are equivalent (Duc, Dziembowski, Faust 2014)



Value x encoded by $t+1$-tuple of prob. values $(x_0 \ldots x_t)$ s.t.

- $x_0, \ldots, x_t$ are i.i.d. w.r.t. to uniform distribution
- $x = x_0 + \ldots + x_t$

Prior work

- Moss, Oswald, Page and Tunstall (2012)
- Bayrak, Regazzoni, Novo and Ienne (2013)
- Eldib, Wang and Schaumont (2014)

Limited to low orders, does not compose well

Probing security, formally

Program c is secure at order t iff

- every set of observations of size $\leq t$ can be simulated with at most $\leq t$ shares from each input;
- every set of observations of size $d \leq t$ can be simulated with at most $\leq d$ shares from each input
- given two equivalent inputs, the joint distributions for a set of observations of size $\leq t$ are equal

Simplified case

Let $f : A_1 \times A_2 \to B$. The following are equivalent:

- there exists $g : A_2 \to B$ s.t. $f(a_1, a_2) = g(a_2)$ for every a_1, a_2
- $f(a_1, a_2) = f(a'_1, a_2)$ for every a_1, a'_1, a_2

MaskVerif

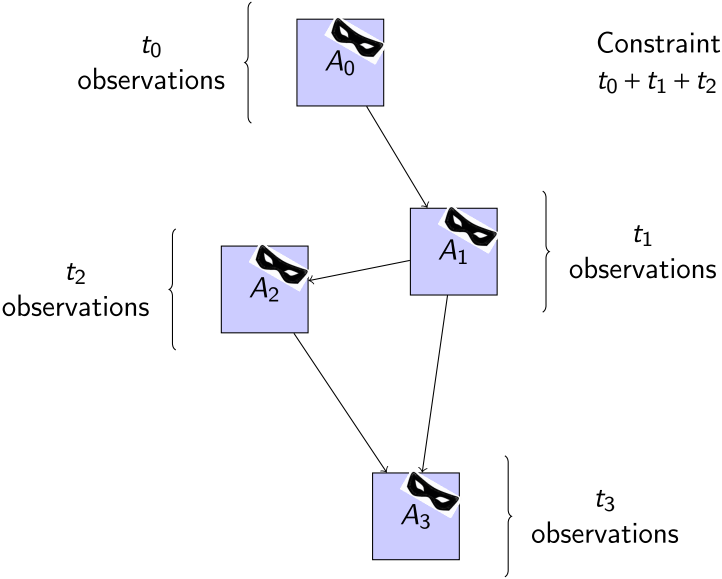
- Check probabilistic non-interference for large sets
- Works well in practice

Reference	Target	# tuples	Security	Complexity # sets	time (s)
First-Order Masking					
FSE13	full AES	17,206	✔	3,342	128
MAC-SHA3	full Keccak-f	13,466	✔	5,421	405
Second-Order Masking					
RSA06	Sbox	1,188,111	✔	4,104	1.649
CHES10	Sbox	7,140	1^{st}-order flaws (2)	866	0.045
CHES10	AES KS	23,041,866	✔	771,263	340,745
FSE13	2 rnds AES	25,429,146	✔	511,865	1,295
FSE13	4 rnds AES	109,571,806	✔	2,317,593	40,169
Third-Order Masking					
RSA06	Sbox	2,057,067,320	3^{rd}-order flaws (98,176)	2,013,070	695
FSE13	Sbox(4)	4,499,950	✔	33,075	3.894
FSE13	Sbox(5)	4,499,950	✔	39,613	5.036
Fourth-Order Masking					
FSE13	Sbox (4)	2,277,036,685	✔	3,343,587	879
Fifth-Order Masking					
CHES10	⊙	216,071,394	✔	856,147	45

MaskComp

- Compositional security notion
- Fully automated type-based information flow analysis (using abstract sets with cardinality constraints)
- Type-driven automated insertion of (SNI) refresh gadgets
- used to mask AES, Keccak, Simon, Speck at high orders
- generated code is reasonably fast, e.g. AES masked at order 7 is $\sim 100\times$ slower than unmasked code

Composition



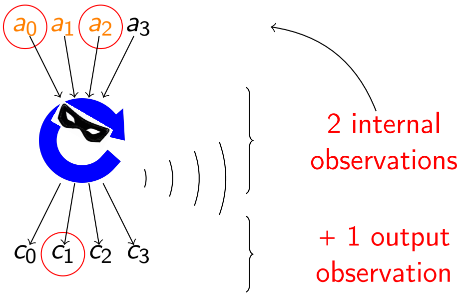
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

Strong non-interference

show that any set of t intermediate variables with

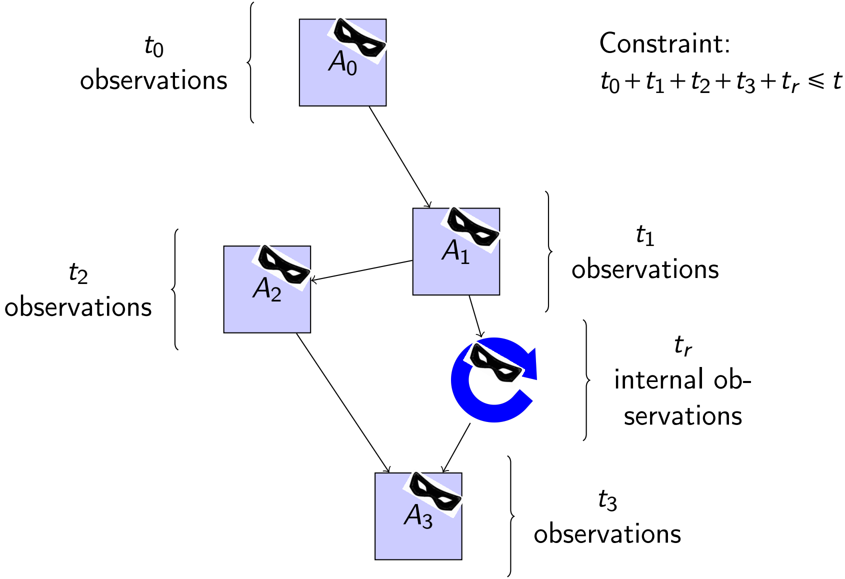
- t_1 on internal variables
- $t_2 = t - t_1$ on the outputs

can be simulated with at most t_1 shares of each input



- Several gadgets are strong non-interfering
- Extended MaskVerif to check SNI

Secure Composition



Status

- Automated synthesis of refreshing gadgets
- Conversion between boolean and arithmetic masking

- Many simulation-based notions of security are equivalent to information flow notions. Language-based techniques apply
- Active attacks (e.g. fault injections) is adversarial program repair. Syntax-guided program synthesis applies

Summary

Foundations and tools for high-assurance cryptography

- Provable security
- Practical cryptography
- Reducing the gap between security proofs and implementations

Many exciting directions

- Automation (lattice-based crypto, etc)
- High-speed implementations (Jasmin)
- Language-based methods for information-theoretic security
- Synthesis (Hoang, Katz, Malozemoff 2015, Carmer, Rosulek 2016)
- Quantum cryptography