Symbolic methods applied to the automation of computational proofs

Charlie Jacomme supervised by Hubert Comon-Lundh & Steve Kremer

November 20,2017

LSV, INRIA Nancy

Introduction to security

Security ?

Why is computer security important ?

Security ?

Why is computer security important ?

Stuxnet

Security ?

Why is computer security important ?

Stuxnet - Took control of Iranian nuclear power plants

Security ?

Boom !

Security ?

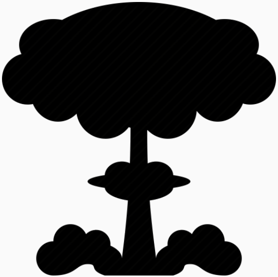
Ok, but why is the security of everyday services important ?
Mails, Facebook, Twitter, ...

Security ?

Ok, but why is the security of everyday services important ?

Mails, Facebook, Twitter, ...

Security ?



Boom !

Security ?

Ok, but why is my security important ?

- CB card

Security ?

Ok, but why is my security important ?

- CB card
- Bank statements
- E-mails
- Internet search history
- ...

Security ?

Ok, but why is my security important ?

- CB card
- Bank statements
- E-mails
- Internet search history
- ...

$\hookrightarrow$ Should I care if a company or a government can read my mails ?

Security ?

Boom !

Security ?

What an attacker could learn about you:

- Do you cheat your spouse ?
- Are you homosexual ? Your son ?
- Are you in need of money ? Are you sick ?

Security ?

What an attacker could learn about you:

- Do you cheat your spouse ?
- Are you homosexual ? Your son ?
- Are you in need of money ? Are you sick ?

It matters!

- Blackmail and corruption
- Commercial targeting
- Harassment and segregation
- Freedom of speech

Conclusion of the Introduction

We want security !

Conclusion of the Introduction

We want formal proofs of security !

Symbolic model

Proofs by saturation

1. Define exactly which operations the attacker can perform.
2. Define the security of our protocol/scheme.
3. Try all possible attacker actions, until we:
    - either break security,
    - or get a security proof.

Proofs by saturation

1. Define exactly which operations the attacker can perform.
2. Define the security of our protocol/scheme.
3. Try all possible attacker actions, until we:
   - either break security,
   - or get a security proof.

Realm

- Messages are abstract terms: $enc(message, sk)$
- Equationnal theory captures the attacker power:

$$dec(enc(m, sk), sk)) = m$$

- The attacker can intercept everything sent over the network

A symbolic method example

Deducibility
Given a set of messages, can an attacker deduce a secret ?

A symbolic method example

Deducibility
Given a set of messages, can an attacker deduce a secret ?

Example

$$g^{x \cdot y}, x, g^{y^2} \vdash^? g^{y^2+y}$$

A symbolic method example

Deducibility
Given a set of messages, can an attacker deduce a secret ?

Example

$$g^{x \cdot y}, x, g^{y^2} \vdash^? g^{y^2+y}$$

$$g^{x \cdot y}$$

Deducibility
Given a set of messages, can an attacker deduce a secret ?

Example

$$g^{x \cdot y}, x, g^{y^2} \vdash^? g^{y^2+y}$$

$$(g^{x \cdot y})^{\frac{1}{x}}$$

A symbolic method example

Deducibility
Given a set of messages, can an attacker deduce a secret ?

Example

$$g^{x \cdot y}, x, g^{y^2} \vdash^? g^{y^2+y}$$

$$(g^{x \cdot y})^{\frac{1}{x}} g^{y^2}$$

A symbolic method example

Deducibility
Given a set of messages, can an attacker deduce a secret ?

Example

$$g^{x \cdot y}, x, g^{y^2} \vdash^? g^{y^2+y}$$

$$(g^{x \cdot y})^{\frac{1}{x}} g^{y^2} = g^{y^2+y}$$

Computational model

Proof by reductions

1. Assume that some problem is computationally difficult
2. Define the security of our protocol/scheme
3. Show that if one can break the security, one can break the difficult problem

Proof by reductions

1. Assume that some problem is computationally difficult
2. Define the security of our protocol/scheme
3. Show that if one can break the security, one can break the difficult problem

Realm

- Messages are bitstrings
- Attackers are any PPT algorithm/TM

Computational vs Symbolic

Fight !

Symbolic model

- Network controlled by the attacker
- Primitives are idealized

Computational model

- Network controlled by the attacker
- Arbitrary PPT attacker

Fight !

Symbolic model

- Network controlled by the attacker
- Primitives are idealized

✓ Many automated proofs

× No proofs by hand

× Missed attacks

Computational model

- Network controlled by the attacker
- Arbitrary PPT attacker

× Few automated proofs

× Hand made proofs hard to check

✓ Stronger proofs

$\hookrightarrow$ Our focus : use technics from symbolic models to improve automation in the computational model

A formal framework for computational proofs

Game equivalence

Goal example:

$$\forall \mathcal{A}.\ a, b : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^{ab}) \simeq a, b, c : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^c)$$

Game equivalence

Arbitraty PPT TM

Goal example:

$$\forall \mathcal{A}.\ a, b : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^{ab}) \simeq a, b, c : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^c)$$

Game equivalence

Arbitraty PPT TM

Goal example:

$$\forall \mathcal{A}.\ \textcolor{red}{a, b} : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab}) \simeq a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c)$$

Randomly sampled over $\mathbb{F}_q$

Game equivalence

Arbitraty PPT TM

Goal example:

$$\forall \mathcal{A}.\ a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab}) \simeq a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c)$$

Randomly sampled over $\mathbb{F}_q$

Equality of distribution

Game equivalence

Arbitraty PPT TM

Goal example:

$$\forall \mathcal{A}.\ a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab}) \simeq a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c)$$

Randomly sampled over $\mathbb{F}_q$

Equality of distribution

$\hookrightarrow$ The attacker cannot distinguish between the two inputs

Reduction example

The DDH assumption

$$\forall \mathcal{A}.\ (a, b : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^c))$$

Reduction example

The DDH assumption

$$\forall \mathcal{A}.\ (a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c))$$

The simulator
We can replace any $\mathcal{A}$ by:

$$B(\mathcal{A})(e_1, e_2, e_3) := d : \mathbb{F}_q, \mathcal{A}(e_1, e_2, g^d, e_3^d)$$

Reduction example

The DDH assumption

$$\forall \mathcal{A}.\ (a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c))$$

The simulator
We can replace any $\mathcal{A}$ by:

$$B(\mathcal{A})(e_1, e_2, e_3) := d : \mathbb{F}_q, \mathcal{A}(e_1, e_2, g^d, e_3^d)$$

The result

$$\forall \mathcal{A}.\ (a, b, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^d, g^{abd})) \simeq (a, b, c, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^d, g^{cd}))$$

Reduction example

The DDH assumption

$$\forall \mathcal{A}.\ (a, b : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^c))$$

The simulator
We can replace any $\mathcal{A}$ by:

$$B(\mathcal{A})(e_1, e_2, e_3) := d : \mathbb{F}_q, \mathcal{A}(e_1, e_2, g^d, e_3^d)$$

The result

$$\forall \mathcal{A}.\ (a, b, d : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^d, g^{abd})) \simeq (a, b, c, d : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^d, g^{cd}))$$

$\hookrightarrow$ We want to do this in reverse, i.e build the B

Automated construction of simulators

The problem

Question
Given an assumption and a goal, can we automatically find B?

Example

Simulator validity

Assumption: $\forall \mathcal{A}.\ (a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c))$

Goal: $a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{abc}) \simeq a, b, c, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{dc})$

Example

Simulator validity

Assumption: $\forall \mathcal{A}.\ (a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c))$

Goal: $a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{abc}) \simeq a, b, c, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{dc})$

Potential simulator B

Example

Simulator validity

Assumption: $\forall \mathcal{A}.\ (a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c))$

Goal: $a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{abc}) \simeq a, b, c, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{dc})$

Potential simulator B

The question
Given (g^a, g^b, g^{ab}), is it possible to compute (g^a, g^b, g^c, g^{abc}) ?

Example

Simulator validity

Assumption: $\forall \mathcal{A}.\ (a, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab})) \simeq (a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c))$

Goal: $a, b, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{abc}) \simeq a, b, c, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{dc})$

Potential simulator B

The question

Given (g^a, g^b, g^{ab}), is it possible to compute (g^a, g^b, g^c, g^{abc}) ?

$\hookrightarrow$A deducibility problem

Main idea

Left hand side of an assumption :

$$x_1, ..., x_n : \mathbb{F}_q.\mathcal{A}(e_1, .., e_k)$$

Left hand side of a goal:

$$x_1, ..., x_n, x_{n+1} ..., x_{n+k} : \mathbb{F}_q.\mathcal{A}(t_1, \ldots, t_l)$$

Check, if for all terms t_i, $1 \leq i \leq l$:

$$e_1, ..., e_k \vdash t_i$$

Correctness of using deducibility

Disadvantage
Something not deducible in the symbolic word might be deducible in the computational world.

$$enc(a, sk), enc(b, sk) \nvdash enc(a + b, sk)$$

Advantage
If something is deducible in the symbolic world, it is always deducible.

$\hookrightarrow$ We may find valid simulators using deducibility.

Option 1
Use existing results in the symbolic model for simple theories.

Using deducibility in practice

Option 1
Use existing results in the symbolic model for simple theories.
$\hookrightarrow$ provides fast automation.

- In case of success, we can construct a simulator
- If it fails, we don't know

Using deducibility in practice

Option 1
Use existing results in the symbolic model for simple theories.

$\hookrightarrow$ provides fast automation.

- In case of success, we can construct a simulator
- If it fails, we don't know

Option 2
Extend the symbolic technics to more faithful theories.

Using deducibility in practice

Option 1
Use existing results in the symbolic model for simple theories.

$\hookrightarrow$ provides fast automation.

- In case of success, we can construct a simulator
- If it fails, we don't know

Option 2
Extend the symbolic technics to more faithful theories.

$\hookrightarrow$ provides slower but complete automation.

Contributions

Existing work

- Deducibility only for polynomials of degree one in the exponent, without axioms (e,g $a \neq 0$)
- AutoGnP [Barthe et al, CCS15] used heuristics to construct simulators

Contributions

Existing work

- Deducibility only for polynomials of degree one in the exponent, without axioms (e,g $a \neq 0$)
- AutoGnP [Barthe et al, CCS15] used heuristics to construct simulators

Contributions

- Axioms ($a \neq 0$)
- Bilinear maps
- Any polynomials in the exponent
- Matrices

Symbolic Proofs for Lattice-Based Cryptography, CCS18

G. Barthe, X. Fan, J. Gancher, B. Gregoire, C. Jacomme, E. Shi

Conclusions

Conclusion regarding the presented work

A complete procedure

Given an assumption and a goal, we provide a complete procedure to decide if the assumption can be applied.

Conclusion regarding the presented work

A complete procedure
Given an assumption and a goal, we provide a complete procedure to decide if the assumption can be applied.

WIP
Use symbolic methods (deducibility, static equivalence, unification):

- to automatize more complex crypto proofs (RND rule)
- to verify masking schemes
- to handle multistage games, oracle games, ...

Conclusion regarding the presented work

A complete procedure

Given an assumption and a goal, we provide a complete procedure to decide if the assumption can be applied.

WIP

Use symbolic methods (deducibility, static equivalence, unification):

- to automatize more complex crypto proofs (RND rule)
- to verify masking schemes
- to handle multistage games, oracle games, ...

Conclusion regarding my PhD

Multiple projects in parallel

Multiple projects in parallel

Try to get the best of both worlds:

- Use symbolic methods to enhance automation in the computational world.

  G. Barthe, B. Gregoire, S. Kremer, P-Y.Strub

- Composing proofs of security in the computational world[1].

  H. Comon-Lundh, G. Scerri

- Case studies[2] in the symbolic world, as exhaustive as possible.

  S. Kremer

[1] prove the security of big protocols by only proving its components.

[2] multi-factor authentication protocols, SSH