Symbolic methods in computational cryptography proofs

G. Barthes, B.Grégoire, Charlie Jacomme, S. Kremer, P-Y. Strub

26 June, 2019

LSV (ENS Paris-Saclay) and LORIA (INRIA Nancy)

Introduction

Secure protocols

Security proofs

- Precise definitions of security;
- precise modelling of the protocol;
- clear assumptions.

Secure protocols

Security proofs

- Precise definitions of security;
- precise modelling of the protocol;
- clear assumptions.

Many, many, many security proofs in the computational model.

Secure protocols

Security proofs

- Precise definitions of security;
- precise modelling of the protocol;
- clear assumptions.

Many, many, many security proofs in the computational model.

So, are we happy ?

What's wrong with cryptographic proofs ?

M. Bellare and P. Rogaway, 2004-2006

In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor.

What's wrong with cryptographic proofs ?

M. Bellare and P. Rogaway, 2004-2006
In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor.

S. Halevi, 2005
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)

What's wrong with cryptographic proofs ?

M. Bellare and P. Rogaway, 2004-2006
In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor.

S. Halevi, 2005
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)

A classical example: RSA-OAEP
From 1994 to 2010, one proof, 5 different papers.

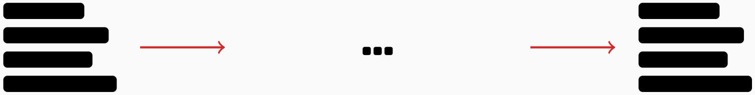
A new hope

V. Shoup, 2004

Security proofs in cryptography may be organized as sequences of games [...] this can be a useful tool in taming the complexity of security proofs that might otherwise become so messy, complicated, and subtle as to be nearly impossible to verify

A new hope

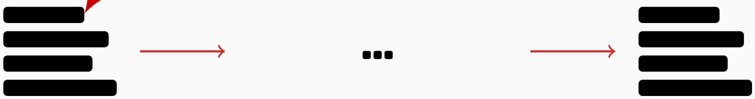
Game sequences



A new hope

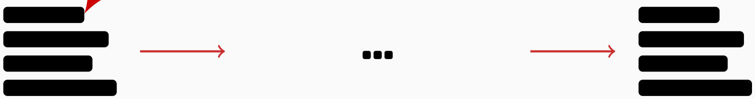
Realistic model of the protocol.

Game sequences



Realistic model of the protocol.

Game sequences

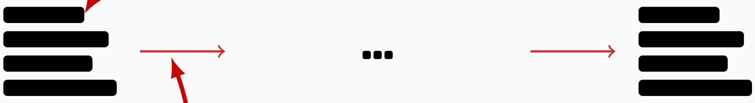


Idealized protocol, where security is clear.

A new hope

Realistic model of the protocol.

Game sequences



Small, verifiable, transformation.

Idealized protocol, where security is clear.

A new hope

Game sequences
Proofs should be easily verifiable, because only based on small transformations.

A new hope

Game sequences
Proofs should be easily verifiable, because only based on small transformations.

So, are we happy ?

A new hope

Game sequences
Proofs are still long and difficult to verify entirely for concrete schemes.

A new hope

Game sequences
Proofs are still long and difficult to verify entirely for concrete schemes.

- but this kind of proof is suited for computer-aided verification.

A new new hope

Mechanized provers
CryptoHol, CryptoVerif, Easycrypt, FCF . . .

A new new hope

Mechanized provers
CryptoHol, CryptoVerif, Easycrypt, FCF ...

Easycrypt
An interactive prover to write formal proofs through game sequences.

A new new hope

Mechanized provers
CryptoHol, CryptoVerif, Easycrypt, FCF ...

Easycrypt
An interactive prover to write formal proofs through game sequences.

So, are we happy ?

The current challenge

Intuition **VS** **EasyCrypt**

The current challenge

Intuition VS EasyCrypt

The current challenge

Intuition **VS** **EasyCrypt**

Our general goal

Automation
Reduce distance between pen and paper proofs and Easycrypt proofs.

Our general goal

Automation
Reduce distance between pen and paper proofs and Easycrypt proofs.

$\hookrightarrow$ automate some game transformations

Our general goal

Automation
Reduce distance between pen and paper proofs and Easycrypt proofs.

$\hookrightarrow$ automate some game transformations

Game transformations
Three important ingredients:

- Uniformity
- Independence
- Equivalence of distribution

Uniformity
Does a message follow the uniform distribution ?

$\hookrightarrow$ the attacker learns nothing

Computational proofs

Uniformity
Does a message follow the uniform distribution ?

$\hookrightarrow$ the attacker learns nothing

Independence (non-interference)
Does a message depend on the distribution of some secret ?

$\hookrightarrow$ no information leakage about the secret

Computational proofs

Uniformity
Does a message follow the uniform distribution ?

$\hookrightarrow$ the attacker learns nothing

Independence (non-interference)
Does a message depend on the distribution of some secret ?

$\hookrightarrow$ no information leakage about the secret

Equivalence
Do two messages have the same probability distribution ?

$\hookrightarrow$ same attacker behaviour

Goal

Precise goal

Decide uniformity, independence and equivalence for simple programs.

Goal

Precise goal
Decide uniformity, independence and equivalence for simple programs.

Simple programs ?

- **inputs/outputs**
- **datatypes** (booleans/bitstrings, $\mathbb{F}_q$, DH exponentiation)
- **constructs** (random sampling, conditionals, bindings)

An example

Easycrypt snippet: Cramer Shoup Key generation

$$x \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$$
$$y, z \xleftarrow{\$} \mathbb{F}_q$$
$$gx, gy, gz \leftarrow g^x, g^y, g^z$$
$$x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{F}_q$$
$$g_1, a, a_1 \leftarrow gx, gy, gz$$
$$k \xleftarrow{\$} dk$$
$$e \leftarrow g^{x1} * g_1^{x2}$$
$$f \leftarrow g^{y1} * g_1^{y2}$$
$$h \leftarrow g^{z1} * g_1^{z2}$$
$$\text{return } pk \leftarrow (k, g, g\_, e, f, g)$$
$$\text{return } sk \leftarrow (k, g, g\_, x_1, x_2, y_1, y_2, z_1, z_2)$$

Easycrypt snippet: Cramer Shoup Key generation

Uniform sampling in a finite field.

$x \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$

$y, z \xleftarrow{\$} \mathbb{F}_q$

$gx, gy, gz \leftarrow g^x, g^y, g^z$

$x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{F}_q$

$g_1, a, a_1 \leftarrow gx, gy, gz$

$k \xleftarrow{\$} dk$

$e \leftarrow g^{x1} * g_1^{x2}$

$f \leftarrow g^{y1} * g_1^{y2}$

$h \leftarrow g^{z1} * g_1^{z2}$

return $pk \leftarrow (k, g, g\_, e, f, g)$

return $sk \leftarrow (k, g, g\_, x_1, x_2, y_1, y_2, z_1, z_2)$

Easycrypt snippet: Cramer Shoup Key generation

$x \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$

$y, z \xleftarrow{\$} \mathbb{F}_q$

$gx, gy, gz \leftarrow g^x, g^y, g^z$

$x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{F}_q$

$g_1, a, a_1 \leftarrow gx, gy, gz$

$k \xleftarrow{\$} dk$

$e \leftarrow g^{x1} * g_1^{x2}$

$f \leftarrow g^{y1} * g_1^{y2}$

$h \leftarrow g^{z1} * g_1^{z2}$

return $pk \leftarrow (k, g, g\_, e, f, g)$

return $sk \leftarrow (k, g, g\_, x_1, x_2, y_1, y_2, z_1, z_2)$

Uniform sampling in a finite field.

Exponentiation in a group.

Easycrypt snippet: Cramer Shoup Key generation

$x \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$

$y, z \xleftarrow{\$} \mathbb{F}_q$

$gx, gy, gz \leftarrow g^x, g^y, g^z$

$x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{F}_q$

$a, a_1 \leftarrow gx, gy, gz$

$k \leftarrow dk$

$e \leftarrow g^{x1} * g_1^{x2}$

$f \leftarrow g^{y1} * g_1^{y2}$

$h \leftarrow g^{z1} * g_1^{z2}$

return $pk \leftarrow (k, g, g\_, e, f, g)$

return $sk \leftarrow (k, g, g\_, x_1, x_2, y_1, y_2, z_1, z_2)$

Uniform sampling in a finite field.

Variable assignment.

Exponentiation in a group.

The EasyCrypt goal

Eascrypt snipet:

$x \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$

$y, z \xleftarrow{\$} \mathbb{F}_q$

$gx, gy, gz \leftarrow g^x, g^y, g^z$

$x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{F}_q$

$g\_, a, a\_ \leftarrow gx, gy, gz$

$k \xleftarrow{\$} dk$

$e \leftarrow g^{x1} * g\_^{x2}$

$f \leftarrow g^{y1} * g\_^{y2}$

$h \leftarrow g^{z1} * g\_^{z2}$

return $pk \leftarrow (k, g, g\_, e, f, g)$

return $sk \leftarrow (k, g, g\_, x_1, x_2, y_1, y_2, z_1, z_2)$

The EasyCrypt goal

Eascrypt snipet:

$x \xleftarrow{\$} \mathbb{F}_q \setminus \{\mathbf{0}\}$
$y, z \xleftarrow{\$} \mathbb{F}_q$
$gx, gy, gz \leftarrow g^x, g^y, g^z$
$x_{\mathbf{1}}, x_{\mathbf{2}}, y_{\mathbf{1}}, y_{\mathbf{2}}, z_{\mathbf{1}}, z_{\mathbf{2}} \xleftarrow{\$} \mathbb{F}_q$
$g\_, a, a\_ \leftarrow gx, gy, gz$
$k \xleftarrow{\$} dk$
$e \leftarrow g^{x\mathbf{1}} * g\_^{x\mathbf{2}}$
$f \leftarrow g^{y\mathbf{1}} * g\_^{y\mathbf{2}}$
$h \leftarrow g^{z\mathbf{1}} * g\_^{z\mathbf{2}}$
return $pk \leftarrow (k, g, g\_, e, f, g)$
return $sk \leftarrow (k, g, g\_, x_{\mathbf{1}}, x_{\mathbf{2}}, y_{\mathbf{1}}, y_{\mathbf{2}}, z_{\mathbf{1}}, z_{\mathbf{2}})$

The EasyCrypt goal

Eascrypt snipet:

$x \xleftarrow{\$} \mathbb{F}_q \setminus \{\mathbf{0}\}$
$y, z \xleftarrow{\$} \mathbb{F}_q$
$gx, gy, gz \leftarrow g^x, g^y, g^z$
$x_{\mathbf{1}}, x_{\mathbf{2}}, y_{\mathbf{1}}, y_{\mathbf{2}}, z_{\mathbf{1}}, z_{\mathbf{2}} \xleftarrow{\$} \mathbb{F}_q$
$g\_, a, a\_ \leftarrow gx, gy, gz$
$k \xleftarrow{\$} dk$
$e \leftarrow g^{x\mathbf{1}} * g\_^{x\mathbf{2}}$
$f \leftarrow g^{y\mathbf{1}} * g\_^{y\mathbf{2}}$
$h \leftarrow g^{z\mathbf{1}} * g\_^{z\mathbf{2}}$
return $pk \leftarrow (k, g, g\_, e, f, g)$
return $sk \leftarrow (k, g, g\_, x_{\mathbf{1}}, x_{\mathbf{2}}, y_{\mathbf{1}}, y_{\mathbf{2}}, z_{\mathbf{1}}, z_{\mathbf{2}})$

The attacker sees $pk := (k, g, g^x, g^{x1+x*x_2}, g^{y1+x*y_2}, g^{z1+x*z_2})$

The EasyCrypt goal

Eascrypt snipet:

$x \xleftarrow{\$} \mathbb{F}_q \setminus \{\mathbf{0}\}$
$y, z \xleftarrow{\$} \mathbb{F}_q$
$gx, gy, gz \leftarrow g^x, g^y, g^z$
$x_{\mathbf{1}}, x_{\mathbf{2}}, y_{\mathbf{1}}, y_{\mathbf{2}}, z_{\mathbf{1}}, z_{\mathbf{2}} \xleftarrow{\$} \mathbb{F}_q$
$g\_, a, a\_ \leftarrow gx, gy, gz$
$k \xleftarrow{\$} dk$
$e \leftarrow g^{x\mathbf{1}} * g\_^{x\mathbf{2}}$
$f \leftarrow g^{y\mathbf{1}} * g\_^{y\mathbf{2}}$
$h \leftarrow g^{z\mathbf{1}} * g\_^{z\mathbf{2}}$
return $pk \leftarrow (k, g, g\_, e, f, g)$
return $sk \leftarrow (k, g, g\_, x_{\mathbf{1}}, x_{\mathbf{2}}, y_{\mathbf{1}}, y_{\mathbf{2}}, z_{\mathbf{1}}, z_{\mathbf{2}})$

The attacker sees $pk := (k, g, g^x, g^{x1+x*x_2}, g^{y1+x*y_2}, g^{z1+x*z_2})$

Is pk independent from x_2, y_2 and z_2 ?

The probabilistic question

Does this expression follow the uniform distribution?

$$(k,\ x,\ x_1 + x * x_2,\ x_2,\ y_1 + x * y_2,\ y_2,\ z_1 + x * z_2,\ z_2)$$

An intuitive characterization

Bijections

$$f(u) \simeq u \Leftrightarrow f \text{ is a bijection}$$

An intuitive characterization

Bijections

$$f(u) \simeq u \Leftrightarrow f \text{ is a bijection}$$

$$f(u, v, w) \simeq (u, v, w) \Leftrightarrow f \text{ is a bijection}$$

An intuitive characterization

Is this function a bijection?

$$(k, x, x_1, x_2, y_1, y_2, z_1, z_2) \mapsto$$
$$(k, x, x_1 + x * x_2, x_2, y_1 + x * y_2, y_2, z_1 + x * z_2, z_2)$$

An intuitive characterization

Is this function a bijection?

$$(k, x, x_1, x_2, y_1, y_2, z_1, z_2) \mapsto$$
$$(k, x, x_1 + x * x_2, x_2, y_1 + x * y_2, y_2, z_1 + x * z_2, z_2)$$

- $x_1 + x * x_2$

An intuitive characterization

Is this function a bijection?

$$(k, x, x_1, x_2, y_1, y_2, z_1, z_2) \mapsto$$
$$(k, x, x_1 + x * x_2, x_2, y_1 + x * y_2, y_2, z_1 + x * z_2, z_2)$$

- $x_1 + x * x_2 \; - x$

An intuitive characterization

Is this function a bijection?

$$(k, x, x_1, x_2, y_1, y_2, z_1, z_2) \mapsto$$
$$(k, x, x_1 + x * x_2, x_2, y_1 + x * y_2, y_2, z_1 + x * z_2, z_2)$$

- $x_1 + x * x_2 - x * x_2$

An intuitive characterization

Is this function a bijection?

$$(k, x, x_1, x_2, y_1, y_2, z_1, z_2) \mapsto$$
$$(k, x, x_1 + x * x_2, x_2, y_1 + x * y_2, y_2, z_1 + x * z_2, z_2)$$

- $x_1 + x * x_2 \; - x * x_2 = x_1$

Symbolic methods

Our question
Only from the outputs of the function, can we compute the inputs ?

Symbolic methods

Our question
Only from the outputs of the function, can we compute the inputs ?

Deducibility
From a set of messages, can we compute some secret.

Symbolic methods

Our question
Only from the outputs of the function, can we compute the inputs ?

Deducibility
From a set of messages, can we compute some secret.

$\hookrightarrow$ Use symbolic methods to perform proofs in the computational model.

Correction and completeness

Deducibility

- Can an attacker deduce a secret ?

Correction and completeness

Deducibility

- Can an attacker deduce a secret ?
- Always correct (a symbolic attack is a computational attack)

Correction and completeness

Deducibility

- Can an attacker deduce a secret ?
- Always correct (a symbolic attack is a computational attack)
- Not always computationally complete (may miss attacks).

$\hookrightarrow$ We only need the correction to have a witness of uniformity.

A general Framework

Programs

Variables

- A set $X = (x, y, z, \dots)$ of deterministic variables;
- a set $R = (u, v, w, \dots)$ of random variables.

Programs
A program is a sequence of terms built over $t \in \mathcal{T}(\Sigma, X \uplus R)$.

Programs examples

Examples

- $P(\{x, y\}, \{u\}) = (x + u, y, xy)$
- $P(\{x, y\}, \{u, v, w\}) = (uv + vw + wu + xy)$

Programs examples

Input : x,y

Sample uniformly u

Return $(x+u, y, xy)$

Examples

- $P(\{x,y\},\{u\}) = (x+u, y, xy)$
- $P(\{x,y\},\{u,v,w\}) = (uv+vw+wu+xy)$

Programs examples

Input : x,y

Sample uniformly u

Return $(x + u, y, xy)$

Examples

- $P(\{x, y\}, \{u\}) = (x + u, y, xy)$
- $P(\{x, y\}, \{u, v, w\}) = (uv + vw + wu + xy)$

Input : x,y

Sample uniformly u,v,w

Return $(uv + vw + wu + xy)$

Probabilistic relations

The framework

Terms and Programs: $P_1(X, R) \in \mathcal{T}(\Sigma, X \uplus R)$

$$P(X, R) = P_1(X, R), \ldots, P_k(X, R)$$

Probabilistic relations

The framework

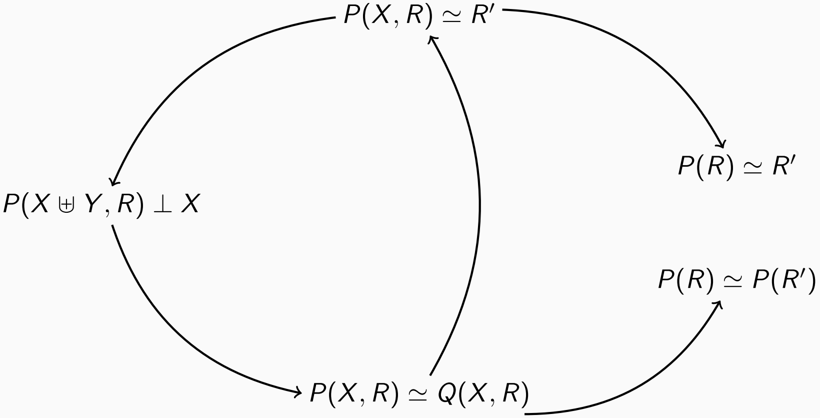
Terms and Programs: $P_1(X,R) \in \mathcal{T}(\Sigma, X \uplus R)$

$$P(X,R) = P_1(X,R), \ldots, P_k(X,R)$$

Relations

Uniformity	$P(X,R) \simeq R$
Independence	$P(X,R) \perp R$
Equivalence	$P(X,R) \simeq Q(X,R)$

Encodings


$P(X, R) \simeq R'$
$P(R) \simeq R'$
$P(X \uplus Y, R) \perp X$
$P(R) \simeq P(R')$
$P(X, R) \simeq Q(X, R)$

Summary of the symbolic abstractions

Deduction
Uniformity for $P(X, R)$ of length $|R| \Leftrightarrow$ Deduction.

Unification and deduction constraints
Equivalence $\Leftrightarrow$ unification and deduction constraints (with private homomorphic symbol).

Static equivalence
Equivalence $\Rightarrow$ static equivalence.

Summary of the symbolic abstractions

Deduction
Uniformity for $P(X, R)$ of length $|R| \Leftrightarrow$ Deduction.

Unification and deduction constraints
Equivalence $\Leftrightarrow$ unification and deduction constraints (with private homomorphic symbol).

Static equivalence
Equivalence $\Rightarrow$ static equivalence.

$\hookrightarrow$ We obtain connections with widely studied questions

The abstraction

Easy to derive heuristics
We can use over and under approximations of the equational theories.

The abstraction

Easy to derive heuristics

We can use over and under approximations of the equational theories.

- If a program follows the uniform distribution when sampling over a ring of characteristic two, it also does when sampling over any $\mathbb{F}_{2^q}$.

The abstraction

Easy to derive heuristics

We can use over and under approximations of the equational theories.

- If a program follows the uniform distribution when sampling over a ring of characteristic two, it also does when sampling over any $\mathbb{F}_{2^q}$.
- If two programs are not equivalent when sampling over $\mathbb{F}_2$, they are not equivalent over a ring of characteristic two.

The abstraction

Modular
There are many combination results for symbolic methods.

The abstraction

Modular
There are many combination results for symbolic methods.

- Easy to add support for free function symbols, or bilinear pairings, or any disjoint equational theories.

Implementation

A generic library

SolvEq

A generic library

SolvEq

- handles deduction and static equivalence in rings and finite fields;

A generic library

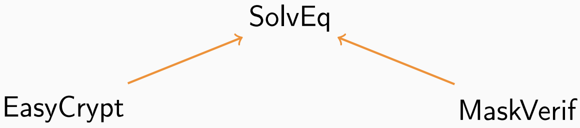
SolvEq

- handles deduction and static equivalence in rings and finite fields;
- procedures/heuristics for uniformity (bijection computations) and independence.

A generic library

- handles deduction and static equivalence in rings and finite fields;
- procedures/heuristics for uniformity (bijection computations) and independence.

A generic library



- handles deduction and static equivalence in rings and finite fields;
- procedures/heuristics for uniformity (bijection computations) and independence.

Sample of Cramer Shoup proofs

```
swap{1} 16 -9;  wp; swap -1; swap -1.
rnd (fun z ⇒  z + G1.w{2} * G1.z2{2})
(fun z ⇒  z - G1.w{2} * G1.z2{2}).
rnd.
wp; swap -1.
rnd (fun z ⇒  z + G1.w{2} * G1.y2{2})
(fun z ⇒  z - G1.w{2} * G1.y2{2}).
rnd.
wp; swap -1.
rnd (fun z ⇒  z + G1.w{2} * G1.x2{2})
(fun z ⇒  z - G1.w{2} * G1.x2{2}).
rnd; wp; rnd; wp.
rnd (fun z ⇒  z / x{1}) (fun z ⇒  z * x{1}) ⇒  /=.
```

Sample of Cramer Shoup proofs

17 tactic calls replaced by a single tactic, with content extracted from cryptographic intuition.

```
rndmatch
    (z1, G1.z, fun z => z + G1.w{2} * G1.z2{2})
    (z2, G1.z2)
    (y1, G1.y, fun z => z + G1.w{2} * G1.y2{2})
    (y2, G1.y2)
    (x1, G1.x, fun z => z + G1.w{2} * G1.x2{2})
    (x2, G1.x2)
    (k , G1.k )
    (z , x , fun z => z / x{1})
    (y , G1.u )
    (x , G1.w ).
```

MaskVerif

Based on a fast heuristic to automatically verify masking schemes (non interference).

- ✓ Very fast;
- ✓ a lot of examples covered.

MaskVerif

Based on a fast heuristic to automatically verify masking schemes (non interference).

- ✓ Very fast;
- ✓ a lot of examples covered.
- × No information when heuristic fails;
- × no negative results;
- × heuristic may fail for simple examples.

MaskVerif

Based on a fast heuristic to automatically verify masking schemes (non interference).

- ✓ Very fast;
- ✓ a lot of examples covered.
- × No information when heuristic fails;
- × no negative results;
- × heuristic may fail for simple examples.

Improvements

- Witnesses of negative results
- New examples not covered by the old heuristic

Conclusion

Final words

The general idea
Use symbolic methods to simplify basic proof steps in the computational model.

Final words

The general idea
Use symbolic methods to simplify basic proof steps in the computational model.

- Link different probabilistic problems;

Final words

The general idea
Use symbolic methods to simplify basic proof steps in the computational model.

- Link different probabilistic problems;
- abstracted into term algebras;

Final words

The general idea

Use symbolic methods to simplify basic proof steps in the computational model.

- Link different probabilistic problems;
- abstracted into term algebras;
- derive algorithms from symbolic methods that are principled, sound and/or complete;

Final words

The general idea
Use symbolic methods to simplify basic proof steps in the computational model.

- Link different probabilistic problems;
- abstracted into term algebras;
- derive algorithms from symbolic methods that are principled, sound and/or complete;
- implement and integrate the resulting algorithms inside existing tools.

Final words

Future work

- automate the application of cryptographic assumptions;
- automate the verification of MPC protocols;
- find an efficient algorithm for general equivalence in finite fields.