CryptoVerif: Mechanising Game-Based Proofs

Part I

Charlie Jacomme

June 06, 2023

Inria Paris

Intro

Who are we?

Benjamin Lipp
benjamin.lipp@mpi-sp.org
(HPKE case study in CryptoVerif)

Who are we?

Benjamin Lipp
benjamin.lipp@mpi-sp.org
(HPKE case study in CryptoVerif)

Charlie Jacomme
charlie.jacomme@inria.fr
(Post-quantum CryptoVerif)

Who are we?

Benjamin Lipp
benjamin.lipp@mpi-sp.org
(HPKE case study in CryptoVerif)

Charlie Jacomme
charlie.jacomme@inria.fr
(Post-quantum CryptoVerif)

Bruno Blanchet
bruno.blanchet@inria.fr
(CryptoVerif's creator, not a fan of travel. . .)

The plan for today

The goal
CryptoVerif: automatically get security guarantees on crypto constructions

The plan for today

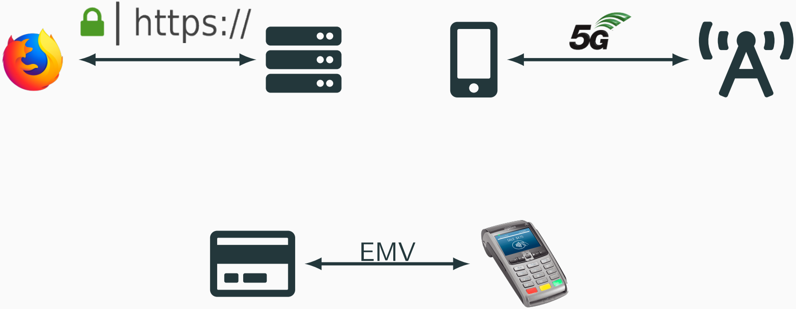
The goal
CryptoVerif: automatically get security guarantees on crypto constructions

Timetable (pessimistic version)

- 9h00-10h30 - Listening/Talking: Context, Motivation, Theory, Demo
- 11h00-12h30 - Doing
- 14h00-15h30 - Listening/Talking: Going further
- 16h00-17h30 - More doing

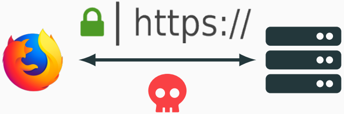
Cryptographic protocols

Cryptographic protocols
Distributed that aims at establishing **secure** communications.

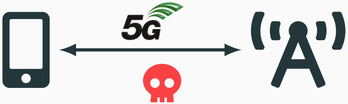


Cryptographic protocols

Cryptographic protocols
Distributed that aims at establishing **secure** communications.



(Sloth – Bhargavan et al. 2015)



(Anonymity – Basin et al. 2018)

(No contact ceiling – Radu et al. 2021)

Security Proofs

To avoid attacks

- a proof rules out all attacks captured by the model;

Security Proofs

To avoid attacks

- a proof rules out all attacks captured by the model;
- a **legal** obligation for e-voting in Switzerland.

Security Proofs

To avoid attacks

- a proof rules out all attacks captured by the model;
- a **legal** obligation for e-voting in Switzerland.

For which attackers?

Symbolic Model

Computational Model

Security Proofs

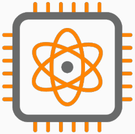
To avoid attacks

- a proof rules out all attacks captured by the model;
- a **legal** obligation for e-voting in Switzerland.

For which attackers?

Symbolic Model

Computational Model



Quantum Computers

Proofs?

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

— Bellare and Rogaway [BR06]

Proofs?

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

— Bellare and Rogaway [BR06]

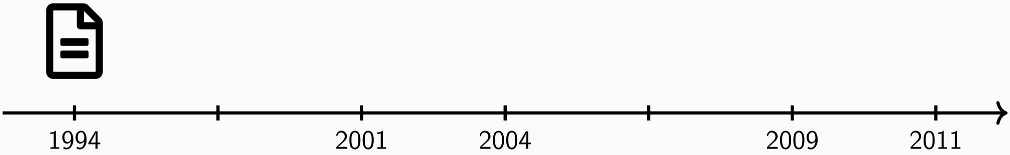
"Some of the reasons for this problem are social (e.g., we mostly publish in conferences rather than journals), but the true cause of it is that our proofs are truly complex."

— Halevi [Hal05]

How to make proofs?

By hand

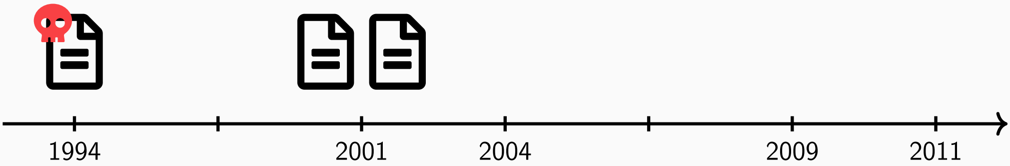
The OAEP proof:



How to make proofs?

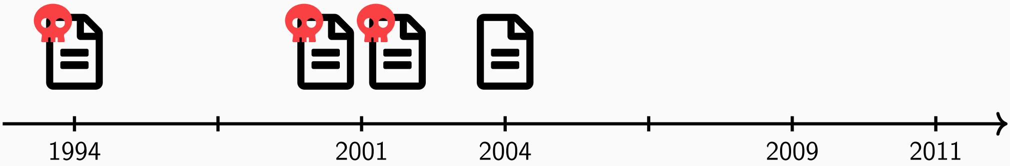
By hand

The OAEP proof:



How to make proofs?

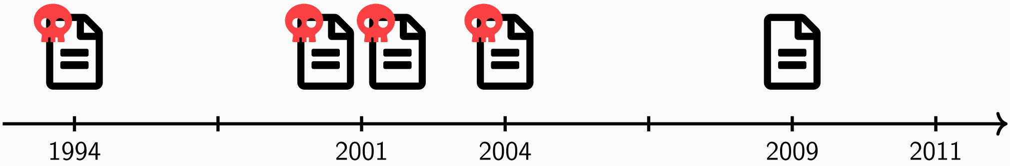
By hand
The OAEP proof:



How to make proofs?

By hand

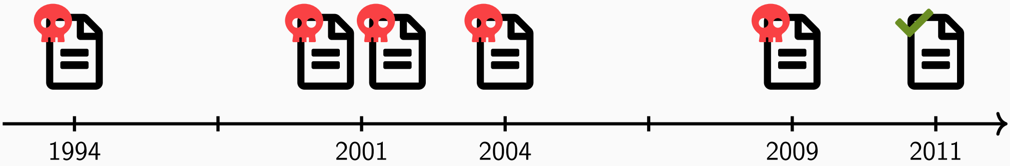
The OAEP proof:



How to make proofs?

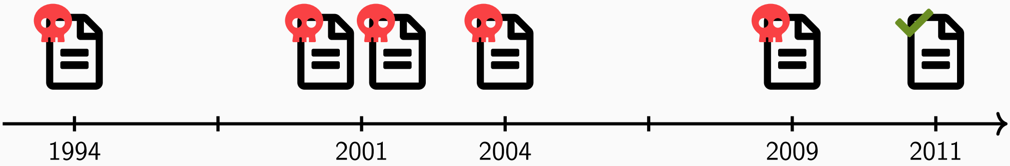
By hand

The OAEP proof:



How to make proofs?

By hand
The OAEP proof:



The solution: computer-aided cryptography
Programs help us do, check, or automate proofs.

(ProVerif, Tamarin, DeepSec, EasyCrypt, CryptoVerif, Squirrel, . . .)

CryptoVerif

- Automated proofs of security
- Works in the classical cryptographic framework
- Used to prove TLS, HPKE, WireGuard, SSH…

Other tools

Symbolic Model

- TAMARIN, DEEPSEC, PROVERIF → high automation, weaker guarantees but works on highly complex protocols

Computational Model

- EASYCRYPT → very low level, no automation, does not scale to protocols
- CRYPTOVERIF → fully automated, both for primitives and protocols
- SQUIRREL → no automation, but scales to more complex protocols

Crypto Proofs

The main proof technique

Indistinguishability
The attacker on the network cannot decide which side it sees

Real World $\approx$ Ideal World

The main proof technique

Indistinguishability
The attacker on the network cannot decide which side it sees

Real World	$\approx$	Ideal World
A tries to send to B some secret		A and B magically share a secret

The main proof technique

Indistinguishability
The attacker on the network cannot decide which side it sees

$$\text{Real World} \qquad \approx_p \qquad \text{Ideal World}$$

$$\max_{\mathcal{A}} | \quad \Pr[\mathtt{Ideal}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathtt{Real}(\mathcal{A}) \Rightarrow 1] \, | \le p$$

The main proof technique

Indistinguishability
The attacker on the network cannot decide which side it sees

$$\text{Real World} \qquad \approx_{p} \qquad \text{Ideal World}$$

Game Hopping

$$\text{Real World} \quad \approx_{p_1} \quad \dots \quad \approx_{p_n} \quad \text{Ideal World}$$

A few game hops

Basics

- $G \approx_0 G$
- $G_1 \approx_{p_1} G_2 \land G_2 \approx_{p_2} G_3 \Rightarrow G_1 \approx_{p_1+p_2} G_3$

A few game hops

Basics

- $G \approx_0 G$
- $G_1 \approx_{p_1} G_2 \wedge G_2 \approx_{p_2} G_3 \Rightarrow G_1 \approx_{p_1+p_2} G_3$

Concrete code examples

- $\boxed{\begin{array}{l} x \leftarrow\$ \{0,1\}^\eta; \\ y \leftarrow\$ \{0,1\}^\eta; \end{array}} \approx_0 \boxed{\begin{array}{l} y \leftarrow\$ \{0,1\}^\eta; \\ x \leftarrow\$ \{0,1\}^\eta; \end{array}}$

A few game hops

Basics

- $G \approx_0 G$
- $G_1 \approx_{p_1} G_2 \wedge G_2 \approx_{p_2} G_3 \Rightarrow G_1 \approx_{p_1+p_2} G_3$

Concrete code examples

- $\boxed{\begin{array}{l} x \leftarrow\!\$\ \{0,1\}^\eta; \\ y \leftarrow\!\$\ \{0,1\}^\eta; \end{array}} \approx_0 \boxed{\begin{array}{l} y \leftarrow\!\$\ \{0,1\}^\eta; \\ x \leftarrow\!\$\ \{0,1\}^\eta; \end{array}}$
- $\boxed{\begin{array}{l} x \leftarrow\!\$\ \{0,1\}^\eta; \\ y \leftarrow\!\$\ \{0,1\}^\eta; \\ \textbf{return}\ x = y \end{array}} \approx_{\frac{1}{2^\eta}} \boxed{\textbf{return}\ \texttt{false}}$

Your first crypto assumption

The IND-CPA game

$$\begin{array}{ll} \underline{\texttt{IND-CPA}_b} & \underline{\texttt{Enc}_b(m_0, m_1)} \\ \quad k \xleftarrow{\$} \mathcal{K} & \quad r \xleftarrow{\$} \{0,1\}^{N_n} \\ \quad \textbf{return } \mathcal{A}^{\texttt{Enc}_b} & \quad \textbf{return } Enc(m_b, k, r) \end{array}$$

Your first crypto assumption

The IND-CPA game

$$\begin{array}{ll} \underline{\texttt{IND-CPA}_b} & \underline{\texttt{Enc}_b(m_0, m_1)} \\ k \xleftarrow{\$} \mathcal{K} & r \xleftarrow{\$} \{0,1\}^{N_n} \\ \textbf{return } \mathcal{A}^{\texttt{Enc}_b} & \textbf{return } Enc(m_b, k, r) \end{array}$$

Encryption security

$$\texttt{IND-CPA}_0 \approx_P \texttt{IND-CPA}_1$$

The main ingredient

Reductions

$$H_1 \approx_p H_2 \Rightarrow C[H_1] \approx_{p+\epsilon(C)} C[H_2]$$

The main ingredient

Reductions

$$H_1 \approx_p H_2 \Rightarrow C[H_1] \approx_{p+\epsilon(C)} C[H_2]$$

Rewriting games
$H_1 \approx_p H_2$ is a **cryptographic assumption**, e.g., big integers are hard to factor:

$$G_1 = C[H_1] \approx_{p+\epsilon(C)} C[H_2]$$

Formalizing game-based proofs?

CryptoVerif modeling

- Implements a kind of programming language for sampling, conditionals, ...
- Allows to define the multiple domains (all bitstrings, keys, ...), called **types**
- Allows to define oracles available to the attacker in parallel or sequentially.

Formalizing game-based proofs?

CryptoVerif modeling

- Implements a kind of programming language for sampling, conditionals, ...
- Allows to define the multiple domains (all bitstrings, keys, ...), called **types**
- Allows to define oracles available to the attacker in parallel or sequentially.

CryptoVerif reasoning
Rewrites games with a set of valid **tactics**, and based on cryptographic assumptions pre-defined in libraries.

Demo!

The IND-CPA game in CryptoVerif: *live-demo-1.ocv*

An IND-CPA variant

$\underline{\texttt{IND-CPA-Z}_b}$

$k \xleftarrow{\$} \mathcal{K}$

return $\mathcal{A}^{\texttt{Enc}_b}$

$\underline{\texttt{Enc}_b(m)}$

$r \xleftarrow{\$} \{0,1\}^{N_n}$

if b **then**

return $Enc(m, k, r)$

else

return $Enc(0^{len(m)}, k, r)$

An IND-CPA variant

$$\begin{array}{l}\underline{\texttt{IND-CPA-Z}_b}\\ \quad k \xleftarrow{\$} \mathcal{K}\\ \quad \textbf{return } \mathcal{A}^{\texttt{Enc}_b}\end{array} \qquad \begin{array}{l}\underline{\texttt{Enc}_b(m)}\\ \quad r \xleftarrow{\$} \{0,1\}^{N_n}\\ \quad \textbf{if } b \textbf{ then}\\ \qquad \textbf{return } Enc(m,k,r)\\ \quad \textbf{else}\\ \qquad \textbf{return } Enc(0^{len(m)},k,r)\end{array}$$

A first CryptoVerif proof?
Assuming that $\texttt{IND-CPA-Z}_0 \approx_P \texttt{IND-CPA-Z}_1$, prove that:

$$\texttt{IND-CPA}_0 \approx_{P+\epsilon} \texttt{IND-CPA}_1$$

Our first proof

Let's simplify
While CryptoVerif can prove arbitrary equivalences, it is easier to prove **secrecy** queries.

Our first proof

Let's simplify
While CryptoVerif can prove arbitrary equivalences, it is easier to prove **secrecy** queries.

$$\underline{\texttt{IND-CPA}}$$
$$b \xleftarrow{\$} \{0,1\} \quad k \xleftarrow{\$} \mathcal{K}$$
$$\textbf{return } \mathcal{A}^{\texttt{Enc}}$$

$$\underline{\texttt{Enc}_b(m_0, m_1)}$$
$$r \xleftarrow{\$} \{0,1\}^{N_n}$$
if b **then**
  return $Enc(m_0, k, r)$
else
  return $Enc(m_1, k, r)$

Our first proof

Let's simplify

While CryptoVerif can prove arbitrary equivalences, it is easier to prove **secrecy** queries.

$\underline{\texttt{IND-CPA}}$

$b \xleftarrow{\$} \{0,1\} \quad k \xleftarrow{\$} \mathcal{K}$

return $\mathcal{A}^{\texttt{Enc}}$

$\underline{\texttt{Enc}_b(m_0, m_1)}$

$r \xleftarrow{\$} \{0,1\}^{N_n}$

if b **then**

return $Enc(m_0, k, r)$

else

return $Enc(m_1, k, r)$

Our goal

Assuming that $\texttt{IND-CPA-Z}_0 \approx_P \texttt{IND-CPA-Z}_1$, prove that b is secret in $\texttt{IND-CPA}$:

Our first proof

Let's simplify
While CryptoVerif can prove arbitrary equivalences, it is easier to prove **secrecy** queries.

$$\begin{array}{l} \underline{\texttt{IND-CPA}} \\ \quad b \xleftarrow{\$} \{0,1\} \quad k \xleftarrow{\$} \mathcal{K} \\ \quad \textbf{return } \mathcal{A}^{\texttt{Enc}} \end{array} \qquad \begin{array}{l} \underline{\texttt{Enc}_b(m_0, m_1)} \\ \quad r \xleftarrow{\$} \{0,1\}^{N_n} \\ \quad \textbf{if } b \textbf{ then} \\ \quad\quad \textbf{return } Enc(m_0, k, r) \\ \quad \textbf{else} \\ \quad\quad \textbf{return } Enc(m_1, k, r) \end{array}$$

Our goal
Assuming that $\texttt{IND-CPA-Z}_0 \approx_P \texttt{IND-CPA-Z}_1$, prove that b is secret in $\texttt{IND-CPA}$:

$$\max_{\mathcal{A}} | \quad \Pr[\texttt{IND-CPA}(\mathcal{A}) \Rightarrow 1] - \Pr[\texttt{IND-CPA}(\mathcal{A}) \Rightarrow 0] - \frac{1}{2} | \leq P + \epsilon$$

Demo!

Cryptoverif

- How is the `IND-CPA-Z` assumption written in the library? *live-demo-2.ocv*
- How to use it to prove the secrecy of b in `IND-CPA`? *live-demo-3.ocv*

Your turn! (soon)

A new primitive

A MAC guarantees the integrity and authenticity of the message because only someone who knows the secret key can compute the MAC.

A new primitive

A MAC guarantees the integrity and authenticity of the message because only someone who knows the secret key can compute the MAC.

Strong UnForgeability under Chosen Message Attacks

$$
\begin{array}{l}
\underline{\texttt{SUF-CMA}_b} \\
\quad k \xleftarrow{\$} \mathcal{K} \\
\quad \mathcal{L} = \emptyset \\
\quad (m, s) \leftarrow \mathcal{A}^{\texttt{Mac}} \\
\quad \textbf{if } b \textbf{ then} \\
\quad\quad \textbf{return } \texttt{verify}(m, k, s) \wedge (m, s) \notin \mathcal{L} \\
\quad \textbf{else} \\
\quad\quad \textbf{return } \mathit{false}
\end{array}
$$

$$
\begin{array}{l}
\underline{\texttt{Mac(m)}} \\
\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \mathsf{mac}(m, k))\} \\
\quad \textbf{return } \mathsf{mac}(m, k)
\end{array}
$$

Encrypt-Then-Mac

Integrity
IND-CPA encryption does not say anything about integrity!

What if $enc(m_1, k, r_1) \oplus enc(m_2, k, r_2) = enc(m_1 \oplus m_2, k, r_2)$?

Solution
We define an authenticated encryption scheme by the **encrypt-then-MAC** construction:

$$enc'(m, (k, mk)) = c1 \,\|\, \mathrm{mac}(c1, mk) \text{ where } c1 = enc(m, k).$$

$$dec'(c1 \| m1, (k, mk)) = \textbf{if } \mathrm{mac}(c1, mk) = m1 \textbf{ then } dec(c1, k) \textbf{ else } \perp$$

The property

Can we prove that decryption only succeeds on honestly produced cyphertext?

$\underline{\texttt{INT-CTXT}}$

$k \xleftarrow{\$} \mathcal{K}$

$\mathcal{L} = \emptyset$

return $\mathcal{A}^{\texttt{Enc},\texttt{Dec}}$

$\underline{\texttt{Enc(m)}}$

$c \xleftarrow{\$} enc'(m, k)$

$\mathcal{L} \leftarrow \mathcal{L} \cup \{c\}$

return c

$\underline{\texttt{DecTest(c)}}$

if $c \in \mathcal{L}$

return *True*

else if $dec'(c, k) \neq \perp$ **then**

Bad

else

return *False*

This morning's goal

CryptoVerif
Under the assumption that *enc* is `IND-CPA`-secure and *mac* is `SUF-CMA`-secure, show that the Encrypt-Then-Mac enc' is `IND-CPA`-secure and `INT-CTXT`-secure.

A few additional CryptoVerif constructs

Tables

Table as global storage
We declare a table as a database where each line stores a tuple of the given type.

```
table tableName(type1, ..., typen).
```

Tables

Table as global storage
We declare a table as a database where each line stores a tuple of the given type.

```
table tableName(type1, ..., typen).
```

Lines can be inserted with

```
insert tableName(value1, ..., valuen);
```

And queries can be made with:

```
get tableName(=value1, var2, ..., varn) in
```

Sequential Oracles

```
let processA(...) =
  O1(...) :=
    ...
    return(...);

  O2(...) :=
    ...
    return(...).
```

```
let processB(...) =
  O3(...) :=
    ...
    return(...);

  O4(...) :=
    ...
    return(...)
```

```
process Ostart() :=
        ...
      return;
      run processA(...) | run processB(...)
```

Pattern matching

Encoding functions
Specific functions can be declared as easily invertible:

```
fun encode(type1, ..., typen) [data].
```

One can then get back the inputs with pattern matching:

```
let encode(var1, ..., varn) = var in
...
```

Reachability query

Events
Events can be defined and raised in games:

```
event bad.
```

```
...; event bad.
```

One can then make an unreachability query:

```
query event(bad) ==> false.
```

And...

That's it!

- A *cheatsheet.ocv* is available.
- You should follow *instructions-practical-session-1.pdf* at: `https://github.com/charlie-j/summer-school-2023/`