Verified
Indifferentiable Hashing into Elliptic Curves

Santiago Zanella Béguelin[1]

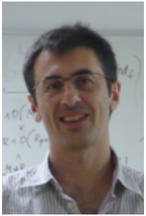
Gilles Barthe[2], Benjamin Grégoire[3],
Sylvain Heraud[3] and Federico Olmedo[2]

Microsoft Research Cambridge[1]

IMDEA Software Institute[2]   INRIA Sophia Antipolis-Méditerranée[3]

2012.03.26
POST 2012

Joint work with



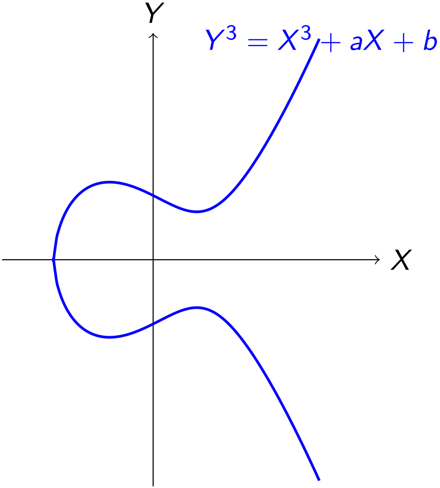
Gilles Barthe

Benjamin Grégoire

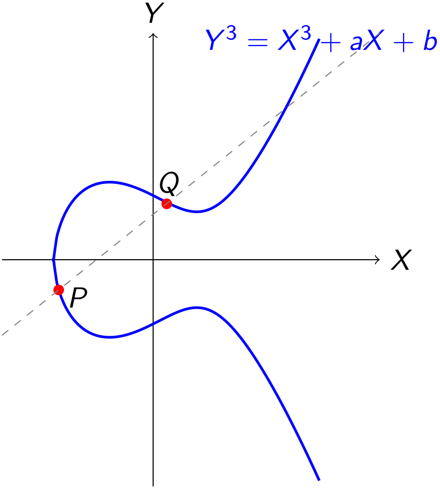
Sylvain Heraud

Federico Olmedo

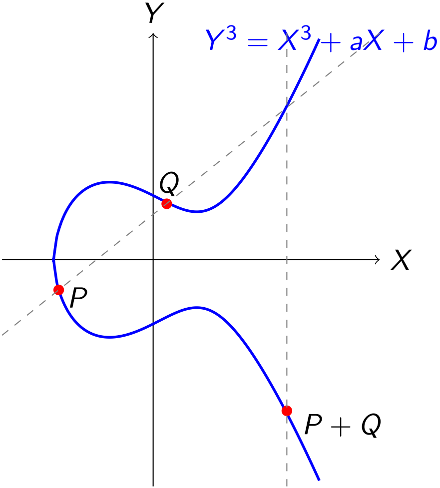
What is an elliptic-curve?



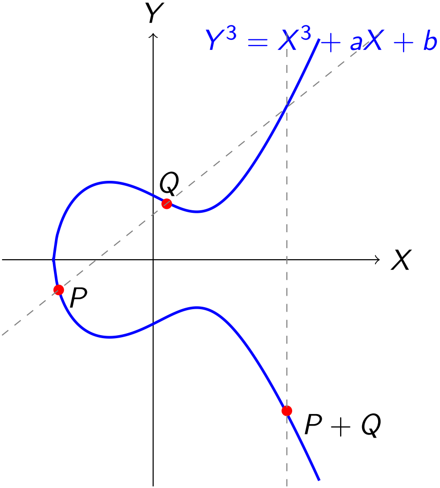
What is an elliptic-curve?



What is an elliptic-curve?



What is an elliptic-curve?



The points in the curve with the point at ∞ form an abelian group

Elliptic Curve Cryptography

Elliptic curve cryptography exploits the algebraic structure of elliptic curves over finite fields

- Based on the hardness of the discrete log problem on EC
- Known methods to solve ECDLP are exponential, compared to sub-exponential for solving RSA
- Achieves same level of security as e.g. RSA but more efficiently (shorter keys—224-bits vs. 2048-bits)

Why it is important to hash into an EC?

- Some useful functionalities can only be achieved efficiently using ECC
- Efficient pairings in Pairing-Based Cryptography are defined on elliptic curves
- Password Authenticated Key Exchange protocols, Identity-Based encryption, signature and signcryption schemes all require hashing into elliptic curves

Boneh-Franklin IBE

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be bilinear pairing and $H : \{0,1\}^* \to \mathbb{G}_1$ a cryptographic hash function [...] The public key associated to an $id \in \{0,1\}^*$ is $Q_{id} = H(id)$ $\longleftarrow$ $\mathbb{G}_1$ is an EC group

Why it is important to hash into an EC?

- Some useful functionalities can only be achieved efficiently using ECC
- Efficient pairings in Pairing-Based Cryptography are defined on elliptic curves
- Password Authenticated Key Exchange protocols, Identity-Based encryption, signature and signcryption schemes all require hashing into elliptic curves

Boneh-Franklin IBE

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be bilinear pairing and $H : \{0,1\}^* \to \mathbb{G}_1$ a cryptographic hash function [...] The public key associated to an $id \in \{0,1\}^*$ is $Q_{id} = H(id)$ $\longleftarrow$ $\mathbb{G}_1$ is an EC group

Why it is difficult to hash (securely) into an EC?

Given a hash function $h : \{0,1\}^* \to \mathbb{F}_p$, how to hash $m \in \{0,1\}^*$ into $EC(\mathbb{F}_p)$?

1. Compute $x = h(m)$. If $\exists y.\ (x, y) \in EC(\mathbb{F}_p)$, return (x, y), otherwise *increment* x and try again.
   - Vulnerable to timing attacks
   - Inefficient
2. Use a determinisitic encoding (e.g. Icart, SWU) $f : \mathbb{F}_p \to EC(\mathbb{F}_p)$: return $f(h(m))$
   - Efficient
   - Differentiable from a random oracle (not surjective / not uniform)

Security proofs of most cryptographic constructions model hash functions as ROs. Implementations are sound only if these hash functions are indifferentiable from a RO

Why it is difficult to hash (securely) into an EC?

Given a hash function $h : \{0,1\}^* \to \mathbb{F}_p$, how to hash $m \in \{0,1\}^*$ into $EC(\mathbb{F}_p)$?

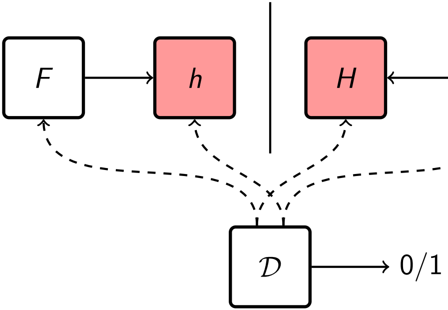
1. Compute $x = h(m)$. If $\exists y.\ (x, y) \in EC(\mathbb{F}_p)$, return (x, y), otherwise *increment* x and try again.
   - Vulnerable to timing attacks
   - Inefficient
2. Use a determinisitic encoding (e.g. Icart, SWU) $f : \mathbb{F}_p \to EC(\mathbb{F}_p)$: return $f(h(m))$
   - Efficient
   - Differentiable from a random oracle (not surjective / not uniform)

Security proofs of most cryptographic constructions model hash functions as ROs. Implementations are sound only if these hash functions are indifferentiable from a RO

Indifferentiability

F with access to a RO h is $(t_{\mathcal{S}}, q, \epsilon)$-indifferentiable from a RO H if

$\exists \mathcal{S}$ that runs in time $t_{\mathcal{S}}$, $\forall \mathcal{D}$ that makes at most q queries,
$\left|\Pr[b \leftarrow \mathcal{D}^{F,h} : b = 1] - \Pr[b \leftarrow \mathcal{D}^{H,\mathcal{S}} : b = 1]\right| \leq \epsilon$

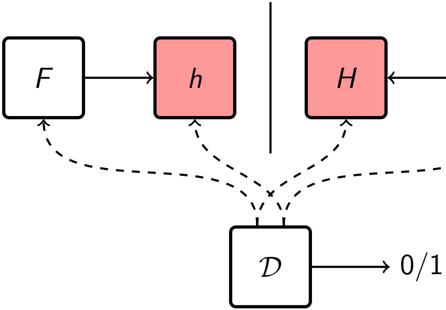


In *any* secure cryptosystem, a random oracle H can be replaced with the construction F, which uses a random oracle h

Indifferentiability

F with access to a RO h is $(t_{\mathcal{S}}, q, \epsilon)$-indifferentiable from a RO H if

$\exists\mathcal{S}$ that runs in time $t_{\mathcal{S}}$, $\forall\mathcal{D}$ that makes at most q queries,
$\left|\Pr[b \leftarrow \mathcal{D}^{F,h} : b = 1] - \Pr[b \leftarrow \mathcal{D}^{H,\mathcal{S}} : b = 1]\right| \leq \epsilon$

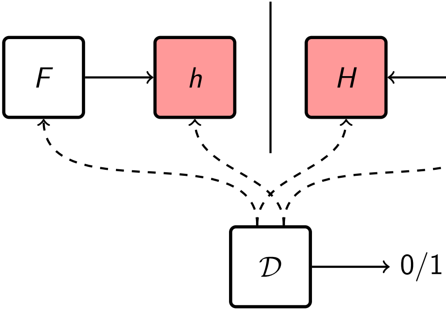


In *any* secure cryptosystem, a random oracle H can be replaced with the construction F, which uses a random oracle h

Indifferentiability

F with access to a RO h is $(t_{\mathcal{S}}, q, \epsilon)$-indifferentiable from a RO H if

$\exists \mathcal{S}$ that runs in time $t_{\mathcal{S}}$, $\forall \mathcal{D}$ that makes at most q queries,
$$\left|\Pr[b \leftarrow \mathcal{D}^{F,h} : b = 1] - \Pr[b \leftarrow \mathcal{D}^{H,\mathcal{S}} : b = 1]\right| \leq \epsilon$$



In *any* secure cryptosystem, a random oracle H into $EC(\mathbb{F}_p)$ can be replaced with the construction F, which uses a random oracle h into $\mathbb{F}_p \times \mathbb{Z}_N$

Indifferentiable Hashing into Elliptic Curves

First indifferentiable construction proposed by Brier et al. in CRYPTO 2010. Given:

- $EC(\mathbb{F}_p) \simeq \mathbb{Z}_N$ with generator g
- Efficiently invertible deterministic encoding $f : \mathbb{F}_p \to EC(\mathbb{F}_p)$
- Random Oracle $h_1 : \{0,1\}^* \to \mathbb{F}_p$
- Random Oracle $h_2 : \{0,1\}^* \to \mathbb{Z}_N$

The construction

$$H(m) = f(h_1(m)) \otimes g^{h_2(m)}$$

is indifferentiable from a random oracle into $EC(\mathbb{F}_p)$

Indifferentiable Hashing into Elliptic Curves

First indifferentiable construction proposed by Brier et al. in CRYPTO 2010. Given:

- $EC(\mathbb{F}_p) \simeq \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$ with generators g_1, g_2
- Efficiently invertible deterministic encoding $f : \mathbb{F}_p \to EC(\mathbb{F}_p)$
- Random Oracle $h_1 : \{0,1\}^* \to \mathbb{F}_p$
- Random Oracle $h_2 : \{0,1\}^* \to \mathbb{Z}_{N_1}$
- Random Oracle $h_3 : \{0,1\}^* \to \mathbb{Z}_{N_2}$

The construction

$$H(m) = f(h_1(m)) \otimes g^{h_2(m)} \otimes g_2^{h_3(m)}$$

is indifferentiable from a random oracle into $EC(\mathbb{F}_p)$

Observation

The group $EC(\mathbb{F}_p)$ is either cyclic or a product of two cyclic groups

The Provable Security paradigm

How can we rigorously prove the indifferentiability of Brier et al. construction?

1. Define an adequate model for the distinguisher $\mathcal{D}$
2. Describe a concrete simulator $\mathcal{S}$
3. Define rigorously the *ideal* ($\mathcal{D}^{H,\mathcal{S}}$) and *real* ($\mathcal{D}^{F,h}$) scenarios
4. Bound the statistical distance between the two scenarios and the running time of $\mathcal{S}$ as a function of the number of queries made by $\mathcal{D}$

Beyond Provable Security: Verifiable Security

How can we formally prove the indifferentiability of Brier et al. construction?

Build a framework to formalize cryptographic proofs

- Provide foundations to cryptographic proofs
- Use a notation as natural as possible for cryptographers
- Automate common reasoning patterns
- Support exact security
- Provide independently and automatically verifiable proofs

CertiCrypt: Language-based cryptographic proofs

Security definitions, assumptions and games are formalized using a probabilistic programming language

pWhile:

$$
\begin{array}{rcll}
\mathcal{C} & ::= & \mathsf{skip} & \text{nop} \\
 & \mid & \mathcal{C};\ \mathcal{C} & \text{sequence} \\
 & \mid & \mathcal{V} \leftarrow \mathcal{E} & \text{assignment} \\
 & \mid & \mathcal{V} \xleftarrow{\$} \mathcal{DE} & \text{random sampling} \\
 & \mid & \mathsf{if}\ \mathcal{E}\ \mathsf{then}\ \mathcal{C}\ \mathsf{else}\ \mathcal{C} & \text{conditional} \\
 & \mid & \mathsf{while}\ \mathcal{E}\ \mathsf{do}\ \mathcal{C} & \text{while loop} \\
 & \mid & \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E}) & \text{procedure call}
\end{array}
$$

$x \xleftarrow{\$} d$: sample the value of x according to distribution d

$$[\![c \in \mathcal{C}]\!] : \mathcal{M} \to \mathsf{Distr}(\mathcal{M})$$

Probabilistic Relational Hoare Logic

Probabilistic extension of Benton's Relational Hoare Logic

Judgments are of the form $c_1 \simeq c_2 : P \Rightarrow Q$, where $P, Q \subseteq \mathcal{M} \times \mathcal{M}$ are binary relations on memories

Definition

$\vDash c_1 \sim c_2 : P \Rightarrow Q \stackrel{\text{def}}{=}$
$$\forall m_1\ m_2,\ m_1\ P\ m_2 \implies [\![c_1]\!]\ m_1\ \mathcal{L}(Q)\ [\![c_2]\!]\ m_2$$
$\mathcal{L}(Q)$ lifts Q to a relation on distributions over memories

Observational equivalence $\vDash c_1 \simeq^I_O c_2$, with $I, O \subseteq \mathcal{V}$ is a special case where:

$$P = \{(m_1, m_2) \mid \forall x \in I\ ,\ \ m_1(x) = m_2(x)\}$$
$$Q = \{(m_1, m_2) \mid \forall x \in O,\ \ m_1(x) = m_2(x)\}$$

From pRHL to probabilities

Assume

$$\models c_1 \sim c_2 : P \Rightarrow Q$$

For all pair of memories m_1, m_2 such that

$$P \;\; m_1 \;\; m_2$$

and events A, B such that

$$Q \implies (A\langle 1\rangle \implies B\langle 2\rangle)$$

we have

$$\Pr[c_1, m_1 : A] \leq \Pr[c_2, m_2 : B]$$

From pRHL to probabilities

Assume

$$\vDash c_1 \sim c_2 : P \Rightarrow Q$$

For all pair of memories m_1, m_2 such that

$$P \ m_1 \ m_2$$

and events A, B such that

$$Q \implies (A\langle 1\rangle \iff B\langle 2\rangle)$$

we have

$$\Pr[c_1, m_1 : A] = \Pr[c_2, m_2 : B]$$

Approximate Observational Equivalence

Simulation-based notions like ϵ-indifferentiability are naturally encoded as approximate equivalence of probabilistic programs

Definition

Approximate Observational Equivalence

$$
\begin{aligned}
&\vDash c_1 \simeq^I_O c_2 \preceq \epsilon \stackrel{\text{def}}{=} \\
&\quad \forall m_1\ m_2\ ,\ m_1\ =_I\ m_2 \implies \\
&\quad \Delta(\llbracket c_1 \rrbracket\ m_1 / =_O, \llbracket c_2 \rrbracket\ m_2 / =_O) \leq \epsilon
\end{aligned}
$$

Can be generalized to a full-fledged Approximate pRHL

Approximate Observational Equivalence

Simulation-based notions like ϵ-indifferentiability are naturally encoded as approximate equivalence of probabilistic programs

Definition

Approximate Observational Equivalence

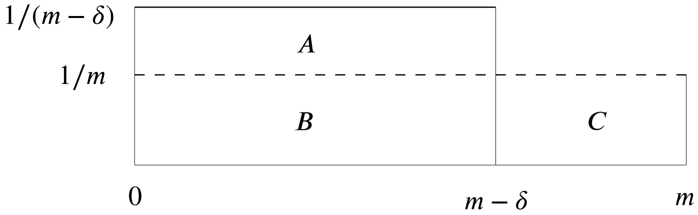
$$
\begin{aligned}
&\vDash c_1 \simeq^I_O c_2 \preceq \epsilon \stackrel{\text{def}}{=} \\
&\quad \forall m_1\ m_2\ ,\ m_1\ =_I\ m_2 \implies \\
&\quad \forall A\ B,\ (m_1 =_O m_2 \implies (A(m_1) \iff B(m_2))) \implies \\
&\quad |\Pr[c_1, m_1 : A] - \Pr[c_2, m_2 : B]| \leq \epsilon
\end{aligned}
$$

Can be generalized to a full-fledged Approximate pRHL

Example: random sampling

$$\frac{\epsilon = \Delta(\mu_1, \mu_2)}{\vDash x \xleftarrow{\$} \mu_1 \simeq^{I}_{I \cup \{x\}} x \xleftarrow{\$} \mu_2 \preceq \epsilon}$$

Sampling from uniform distributions:



$$\vDash x \xleftarrow{\$} \{0, .., m - \delta\} \simeq^{I}_{I \cup \{x\}} x \xleftarrow{\$} \{0, .., m\} \preceq 1/2(A + C) = \delta/m$$

Recap: what we want to prove

Given:

- An elliptic curve group $EC(\mathbb{F}_p) \simeq \mathbb{Z}_N$ with generator g
- An efficiently invertible deterministic encoding $f : \mathbb{F}_p \to EC(\mathbb{F}_p)$
- A Random Oracle $h : \{0,1\}^* \to \mathbb{F}_p \times \mathbb{Z}_N$

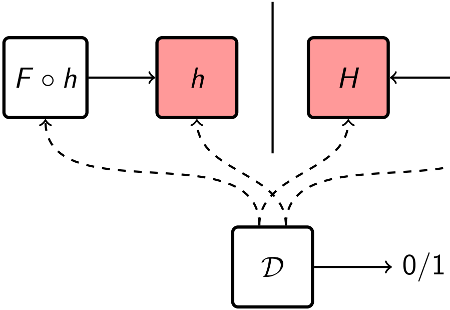
Define

$$F(u, z) \stackrel{\text{def}}{=} f(u) + g^z$$

The construction $F \circ h : \{0,1\}^* \to EC(\mathbb{F}_p)$ is indifferentiable from a random oracle.

Recap: what we want to prove

$\exists \mathcal{S}$ that runs in time $t_{\mathcal{S}}$, $\forall \mathcal{D}$ that makes at most q queries,
$\left|\Pr[b \leftarrow \mathcal{D}^{F \circ h, h} : b = 1] - \Pr[b \leftarrow \mathcal{D}^{H, \mathcal{S}} : b = 1]\right| \leq \epsilon$



Proof sketch

1. We show that an invertible encoding $f : S \to R$ is a *weak encoding*
2. We show that a weak encoding is also an *admissible encoding*
3. We show that an admissible encoding f composed with a random oracle $h : \{0,1\}^* \to S$ is indifferentiable from a random oracle into R

Example: main theorem

Theorem (Indifferentiability)

An ϵ-admissible encoding $f : S \to R$ composed with a random oracle $h : \{0,1\}^* \to S$ is indifferentiable from a random oracle

An ϵ-admissible encoding comes with an efficient inverter $\mathcal{I}_f$ that satisfies:

$$\models r \overset{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r) \simeq^{\emptyset}_{\{s\}} s \overset{\$}{\leftarrow} S \preceq \epsilon$$

We prove first that

$$\models s \overset{\$}{\leftarrow} S;\ r \leftarrow f(s) \simeq^{\emptyset}_{\{r,s\}} r \overset{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r) \preceq 2\epsilon$$

Example: main theorem

Define

$$
\begin{aligned}
c_i &\stackrel{\text{def}}{=} s \xleftarrow{\$} S;\ r \leftarrow f(s) \\
c_f &\stackrel{\text{def}}{=} r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) \\
c_1 &\stackrel{\text{def}}{=} c_i;\ \text{if } s = \bot \text{ then } r \xleftarrow{\$} R \text{ else } r \leftarrow f(s) \\
c_2 &\stackrel{\text{def}}{=} c_f;\ \text{if } s = \bot \text{ then bad} \leftarrow \text{true};\ r \xleftarrow{\$} R \text{ else } r \leftarrow f(s) \\
c_3 &\stackrel{\text{def}}{=} c_f;\ \text{if } s = \bot \text{ then bad} \leftarrow \text{true else } r \leftarrow f(s)
\end{aligned}
$$

The conditional in c_1 is dead-code:

$$\models c_i \simeq^{\emptyset}_{\{r,s\}} c_1$$

Since sequential composition preserves statistical distance:

$$\models c_1 \simeq^{\emptyset}_{\{r,s\}} c_2 \preceq \epsilon$$

Since $\models s \xleftarrow{\$} S \simeq^{\emptyset}_{\{s\}} c_f \preceq \epsilon$,

$$\Pr[c_2 : \text{bad}] = \Pr[s \xleftarrow{\$} S : s \neq \bot] - \Pr[c_f : s \neq \bot] \leq \epsilon$$

$$\models c_2 \simeq^{\emptyset}_{\{r,s\}} c_3 \preceq \epsilon$$

Since the *else* branch in c_3 is dead-code: $\models c_3 \simeq^{\emptyset}_{\{r,s\}} c_f$

Example: main theorem

Define

$$
\begin{array}{ll}
c_i & \stackrel{\text{def}}{=} s \stackrel{\$}{\leftarrow} S;\ r \leftarrow f(s) \\
c_f & \stackrel{\text{def}}{=} r \stackrel{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r) \\
c_1 & \stackrel{\text{def}}{=} c_i;\ \text{if } s = \bot \text{ then } r \stackrel{\$}{\leftarrow} R \text{ else } r \leftarrow f(s) \\
c_2 & \stackrel{\text{def}}{=} c_f;\ \text{if } s = \bot \text{ then bad} \leftarrow \text{true};\ r \stackrel{\$}{\leftarrow} R \text{ else } r \leftarrow f(s) \\
c_3 & \stackrel{\text{def}}{=} c_f;\ \text{if } s = \bot \text{ then bad} \leftarrow \text{true else } r \leftarrow f(s)
\end{array}
$$

The conditional in c_1 is dead-code:

$$\models c_i \simeq^{\emptyset}_{\{r,s\}} c_1$$

Since sequential composition preserves statistical distance:

$$\models c_1 \simeq^{\emptyset}_{\{r,s\}} c_2 \preceq \epsilon$$

Since $\models s \stackrel{\$}{\leftarrow} S \simeq^{\emptyset}_{\{s\}} c_f \preceq \epsilon$,

$$\Pr[c_2 : \text{bad}] = \Pr[s \stackrel{\$}{\leftarrow} S : s \neq \bot] - \Pr[c_f : s \neq \bot] \leq \epsilon$$

$$\models c_2 \simeq^{\emptyset}_{\{r,s\}} c_3 \preceq \epsilon$$

Since the *else* branch in c_3 is dead-code: $\models c_3 \simeq^{\emptyset}_{\{r,s\}} c_f$

Example: main theorem

Define

$$
\begin{array}{ll}
c_i & \stackrel{\text{def}}{=} s \xleftarrow{\$} S;\ r \leftarrow f(s) \\
c_f & \stackrel{\text{def}}{=} r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) \\
c_1 & \stackrel{\text{def}}{=} c_i;\ \text{if } s = \bot \text{ then } r \xleftarrow{\$} R \text{ else } r \leftarrow f(s) \\
c_2 & \stackrel{\text{def}}{=} c_f;\ \text{if } s = \bot \text{ then bad} \leftarrow \text{true};\ r \xleftarrow{\$} R \text{ else } r \leftarrow f(s) \\
c_3 & \stackrel{\text{def}}{=} c_f;\ \text{if } s = \bot \text{ then bad} \leftarrow \text{true else } r \leftarrow f(s)
\end{array}
$$

The conditional in c_1 is dead-code:

$$\vDash c_i \simeq^{\emptyset}_{\{r,s\}} c_1$$

Since sequential composition preserves statistical distance:

$$\vDash c_1 \simeq^{\emptyset}_{\{r,s\}} c_2 \preceq \epsilon$$

Since $\vDash s \xleftarrow{\$} S \simeq^{\emptyset}_{\{s\}} c_f \preceq \epsilon$,

$$\Pr[c_2 : \text{bad}] = \Pr[s \xleftarrow{\$} S : s \neq \bot] - \Pr[c_f : s \neq \bot] \leq \epsilon$$

$$\vDash c_2 \simeq^{\emptyset}_{\{r,s\}} c_3 \preceq \epsilon$$

Since the *else* branch in c_3 is dead-code: $\vDash c_3 \simeq^{\emptyset}_{\{r,s\}} c_f$

Example: main theorem

Define

$$
\begin{array}{ll}
c_i & \stackrel{\mathrm{def}}{=} s \stackrel{\$}{\leftarrow} S;\ r \leftarrow f(s) \\
c_f & \stackrel{\mathrm{def}}{=} r \stackrel{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r) \\
c_1 & \stackrel{\mathrm{def}}{=} c_i;\ \text{if } s = \perp \text{ then } r \stackrel{\$}{\leftarrow} R \text{ else } r \leftarrow f(s) \\
c_2 & \stackrel{\mathrm{def}}{=} c_f;\ \text{if } s = \perp \text{ then bad} \leftarrow \text{true};\ r \stackrel{\$}{\leftarrow} R \text{ else } r \leftarrow f(s) \\
c_3 & \stackrel{\mathrm{def}}{=} c_f;\ \text{if } s = \perp \text{ then bad} \leftarrow \text{true else } r \leftarrow f(s)
\end{array}
$$

The conditional in c_1 is dead-code:

$$\models c_i \simeq^{\emptyset}_{\{r,s\}} c_1$$

Since sequential composition preserves statistical distance:

$$\models c_1 \simeq^{\emptyset}_{\{r,s\}} c_2 \preceq \epsilon$$

Since $\models s \stackrel{\$}{\leftarrow} S \simeq^{\emptyset}_{\{s\}} c_f \preceq \epsilon$,

$$\Pr[c_2 : \text{bad}] = \Pr[s \stackrel{\$}{\leftarrow} S : s \neq \perp] - \Pr[c_f : s \neq \perp] \leq \epsilon$$

$$\models c_2 \simeq^{\emptyset}_{\{r,s\}} c_3 \preceq \epsilon$$

Since the *else* branch in c_3 is dead-code: $\models c_3 \simeq^{\emptyset}_{\{r,s\}} c_f$

Example: main theorem

Game G : $\mathbf{L} \leftarrow \mathsf{nil}$; $b \leftarrow \mathcal{D}(\,)$

Oracle $\mathcal{O}_1(x)$:
if $x \notin \mathsf{dom}(\mathbf{L_1})$ then
$s \xleftarrow{\$} S$; $\mathbf{L_1}(x) \leftarrow s$
return $\mathbf{L_1}(x)$

Oracle $\mathcal{O}_2(x)$:
if $x \notin \mathsf{dom}(\mathbf{L_2})$ then
$s \leftarrow \mathcal{O}_1(x)$; $r \leftarrow f(s)$; $\mathbf{L_2}(x) \leftarrow r$
return $\mathbf{L_2}(x)$

Game G' : $\mathbf{L} \leftarrow \mathsf{nil}$; $b \leftarrow \mathcal{D}(\,)$

Oracle $\mathcal{O}_1(x)$:
if $x \notin \mathsf{dom}(\mathbf{L_1})$ then
$r \leftarrow \mathcal{O}_2(x)$; $s \leftarrow \mathcal{I}_f(r)$; $\mathbf{L_1}(x) \leftarrow s$
return $\mathbf{L_1}(x)$

Oracle $\mathcal{O}_2(x)$:
if $x \notin \mathsf{dom}(\mathbf{L_2})$ then
$r \xleftarrow{\$} R$; $\mathbf{L_2}(x) \leftarrow r$
return $\mathbf{L_2}(x)$

Game G_1 : $\mathbf{L} \leftarrow \mathsf{nil}$; $b \leftarrow \mathcal{A}(\,)$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathsf{dom}(\mathbf{L})$ then
$s \xleftarrow{\$} S$; $r \leftarrow f(s)$; $\mathbf{L}(x) \leftarrow (s, r)$
return $\mathbf{L}(x)$

Game G_2 : $\mathbf{L} \leftarrow \mathsf{nil}$; $b \leftarrow \mathcal{A}(\,)$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathsf{dom}(\mathbf{L})$ then
$r \xleftarrow{\$} R$; $s \leftarrow \mathcal{I}_f(r)$; $\mathbf{L}(x) \leftarrow (s, r)$
return $\mathbf{L}(x)$

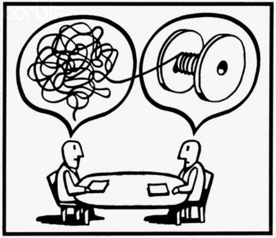
Game $\mathsf{G}_1^{\mathbf{bad}}$: $\mathbf{L} \leftarrow \mathsf{nil}$; $b \leftarrow \mathcal{A}(\,)$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathsf{dom}(\mathbf{L})$ then
if $|\mathbf{L}| < q_1 + q_2$ then
$s \xleftarrow{\$} S$; $r \leftarrow f(s)$
else **bad** $\leftarrow$ true; $s \xleftarrow{\$} S$; $r \leftarrow f(s)$
$\mathbf{L}(x) \leftarrow (s, r)$
return $\mathbf{L}(x)$

Game $\mathsf{G}_2^{\mathbf{bad}}$: $\mathbf{L} \leftarrow \mathsf{nil}$; $b \leftarrow \mathcal{A}(\,)$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathsf{dom}(\mathbf{L})$ then
if $|\mathbf{L}| < q_1 + q_2$ then
$s \xleftarrow{\$} S$; $r \leftarrow f(s)$
else **bad** $\leftarrow$ true; $r \xleftarrow{\$} R$; $s \leftarrow I_f(r)$
$\mathbf{L}(x) \leftarrow (s, r)$
return $\mathbf{L}(x)$

Summary



- Extended CertiCrypt with a novel notion of approximate program equivalence
- First machine-checked security proof of an EC construction
- First machine-checked proof of (exact) indifferentiability

The proof is a *tour-de-force*:

- More than 10,000 original lines of Coq (65k lines in total)
- Approximately 1 man-year effort
- Integrates independently-developed mathematical libraries
- Requires heavy algebraic reasoning

Some directions of research

`http://certicrypt.gforge.inria.fr`

- Generalizations of approximate equivalence to encode DP
- Use approximate equivalence to capture Statistical ZK
- Verifiable proofs of indifferentiability of SHA-3 finalists
- Extend EasyCrypt to reason about approximate equivalence