Verified Indifferentiable Hashing into Elliptic Curves

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

[1] IMDEA Software Institute, Madrid, Spain
{Gilles.Barthe,Federico.Olmedo,Santiago.Zanella}@imdea.org
[2] INRIA Sophia Antipolis-Méditerranée, France
{Benjamin.Gregoire,Sylvain.Heraud}@inria.fr

Abstract. Many cryptographic systems based on elliptic curves are proven secure in the Random Oracle Model, assuming there exist probabilistic functions that map elements in some domain (e.g. bitstrings) onto uniformly and independently distributed points in a curve. When implementing such systems, and in order for the proof to carry over to the implementation, those mappings must be instantiated with concrete constructions whose behavior does not deviate significantly from random oracles. In contrast to other approaches to public-key cryptography, where candidates to instantiate random oracles have been known for some time, the first generic construction for hashing into ordinary elliptic curves indifferentiable from a random oracle was put forward only recently by Brier et al. We present a machine-checked proof of this construction. The proof is based on an extension of the CertiCrypt framework with logics and mechanized tools for reasoning about approximate forms of observational equivalence, and integrates mathematical libraries of group theory and elliptic curves.

1 Introduction

Following an established trend [18], the prevailing methodology for building secure cryptosystems is to conduct a rigorous analysis that proves security under standard hypotheses. Sometimes this analysis is performed assuming that some components of the system have an ideal behavior. However, ideal functionalities are difficult or even impossible to realize, leading to situations where provably secure systems have no secure implementation. An alternative methodology is to devise systems based on constructions that do not deviate significantly from ideal ones, and to account for these deviations in the security analysis. Statistical distance is a natural notion for quantifying the deviation between idealized functionalities and their implementations.

Verifiable security [3,4] is an emerging approach that advocates the use of interactive proof assistants and automated provers to establish the security of cryptographic systems. It improves on the guarantees of provable security by delivering fully machine-checked and independently verifiable proofs. The CertiCrypt framework, built on top of the Coq proof assistant, is one prominent tool that realizes verifiable security by using standard techniques from programming languages and program verification. CertiCrypt is built around the central notion of observational equivalence of probabilistic programs, which unfortunately cannot model accurately other weaker, quantitative,

P. Degano and J.D. Guttman (Eds.): POST 2012, LNCS 7215, pp. 209–228, 2012.

forms of equivalence. As a result, CertiCrypt cannot be used as it is to reason about the statistical distance of distributions generated by probabilistic programs. More generally, the development of quantitative notions of equivalence is quite recent and rather limited; see Section 7 for an account of related work.

One main contribution of this article is the formalization of several quantitative notions of program equivalence and logics for reasoning about them. More specifically, we extend CertiCrypt with the notion of statistical distance and develop a logic to upper bound the distance between distributions generated by probabilistic programs. Moreover, we introduce approximate and conditional variants of observational equivalence and develop equational theories for reasoning about them.

In a landmark article, Maurer et al. [23] introduce the concept of indifferentiability to justify rigorously the substitution of an idealized component in a cryptographic system by a concrete implementation. In a subsequent article, Coron et al. [13] argue that a secure hash function should be indifferentiable from a random oracle, i.e. a perfectly random function. Although the random oracle model has been under fierce criticism [10] and the indifferentiability framework turns out to be weaker than initially believed [16, 25], it is generally accepted that proofs in these models provide some evidence that a system is secure. Not coincidentally, all finalists in the ongoing NIST Cryptographic Hash Algorithm competition have been proved indifferentiable from a random oracle.

Elliptic curve cryptography allows to build efficient public-key cryptographic systems with comparatively short keys and as such is an attractive solution for resource-constrained applications. In contrast to other approaches to public-key cryptography, where candidates to instantiate random oracles have been known for some time, adequate constructions for hashing into ordinary elliptic curves have remained elusive. In 2010, Brier et al. [9] proposed the first generic construction indifferentiable from a random oracle into elliptic curves. This construction is of practical significance since it allows to securely implement elliptic curve cryptosystems. We present a machine-checked and independently verifiable proof of the security of this construction. The proof involves the various notions of equivalence we develop in this paper and is thus an excellent testbed for evaluating the applicability of our methods. Additionally, the proof builds on several large developments (including Théry's formalization of elliptic curves [30] and Gonthier et al. formalization of finite groups [19]) and demonstrates that CertiCrypt blends well with large and complex mathematical libraries, and is apt to support proofs involving advanced algebraic and number-theoretical reasoning.

Organization of the paper. The remainder of the paper is structured as follows. Section 2 provides a brief introduction to CertiCrypt. Section 3 introduces the notion of statistical distance between probabilistic programs and describes programming language techniques to bound it, whereas Sect. 4 defines weak forms of observational equivalence and their associated reasoning principles. Section 5 presents a machine-checked proof of the indifferentiability of a generalization of Brier et al.'s construction from a random oracle into an abelian finite group; its application to elliptic curves is discussed in Sect. 6. We survey prior art and conclude in Sections 7 and 8.

2 An Overview of CertiCrypt

This section provides a brief description of the CertiCrypt framework. We refer the reader to [4] for further details.

2.1 Representation of Distributions

CertiCrypt adopts the monadic representation of distributions proposed by Audebaud and Paulin in [2]. A distribution over a set A is represented as a monotonic, continuous and linear function of type

$$\mathcal{D}(A) \stackrel{\text{def}}{=} (A \to [0,1]) \to [0,1]$$

where $[0,1]$ denotes the unit interval. Intuitively, an element of type $\mathcal{D}(A)$ models the expectation operator of a sub-probability distribution over A. Thus, the probability that a distribution $\mu : \mathcal{D}(A)$ assigns to an event $X \subseteq A$ can be computed by measuring its characteristic function $\mathbb{1}_X$, i.e. $\Pr[\mu : X] \stackrel{\text{def}}{=} \mu(\mathbb{1}_X)$.

2.2 Programming Model

We model games as probabilistic imperative programs with procedure calls. The set of commands $\mathcal{C}$ is defined inductively by the clauses:

$\mathcal{C} ::= \mathsf{skip}$	nop
$\mid \mathcal{V} \leftarrow \mathcal{E}$	deterministic assignment
$\mid \mathcal{V} \stackrel{\$}{\leftarrow} \mathcal{D}\mathcal{E}$	random assignment
$\mid \mathsf{if}\ \mathcal{E}\ \mathsf{then}\ \mathcal{C}\ \mathsf{else}\ \mathcal{C}$	conditional
$\mid \mathsf{while}\ \mathcal{E}\ \mathsf{do}\ \mathcal{C}$	while loop
$\mid \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E})$	procedure call
$\mid \mathcal{C};\ \mathcal{C}$	sequence

where $\mathcal{V}$ is a set of variables tagged with their scope (either local or global), $\mathcal{E}$ is a set of deterministic expressions, and $\mathcal{D}\mathcal{E}$ is a set of expressions that denote distributions from which values can be sampled in random assignments. In the remainder, we let $\mathsf{true} \oplus_\delta \mathsf{false}$ denote the Bernoulli distribution with success probability δ, so that the instruction $x \stackrel{\$}{\leftarrow} \mathsf{true} \oplus_\delta \mathsf{false}$ assigns true to x with probability δ, and we denote by $x \stackrel{\$}{\leftarrow} A$ the instruction that assigns to x a value uniformly chosen from a finite set A.

A program (or game) consists of a command c and an environment E that maps procedure identifiers to their declaration, specifying its formal parameters, its body, and a return expression that is evaluated upon exit. (Although procedures are single-exit, we often write games using explicit return expressions for the sake of readability.) Declarations are subject to well-formedness and well-typedness conditions; these conditions are enforced using the underlying dependent type system of Coq. Procedures corresponding to adversaries are modelled as procedures with unknown code.

Program states (or memories) are dependently typed functions that map a variable of type T to a value in its interpretation $[\![T]\!]$; we let $\mathcal{M}$ denote the set of states. Expressions

have a deterministic semantics: an expression e of type T is interpreted as a function $[\![e]\!] : \mathcal{M} \to [\![T]\!]$. The semantics of a command c in an environment E relates an initial memory to a probability sub-distribution over final memories: $[\![c, E]\!] : \mathcal{M} \to \mathcal{D}(\mathcal{M})$. We often omit the environment when it is irrelevant.

By specializing the above definition of probability $\Pr[\mu : X]$ to programs, we have that the probability $\Pr[G, m : X]$ of an event X in a game G and an initial memory m is given by $[\![G]\!]\, m\, \mathbb{1}_X$. The probability of termination of a game G starting in an initial memory m is given by $\Pr[G, m : \mathsf{true}]$. We say that a game is *lossless* if it terminates with probability 1 independently from the initial memory.

In order to reason about program complexity and define the class of probabilistic polynomial-time computations, the semantics of programs is indexed by a security parameter (a natural number) and instrumented to compute the time and memory cost of evaluating a command, given the time and memory cost of each construction in the expression language. We chose not to make this parameterization explicit to avoid cluttering the presentation.

2.3 Reasoning Tools

CertiCrypt provides several tools for reasoning about games. One main tool is a probabilistic relational Hoare logic. Its judgments are of the form $\models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi$, where G_1 and G_2 are games, and Ψ and Φ are relations over states. We represent relations as first-order formulae over tagged program variables; we use the tags $\langle 1 \rangle$ and $\langle 2 \rangle$ to distinguish between the value of a variable or formula in the left and right-hand side program, respectively.

Formally, a judgment $\models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi$ is valid, iff for all memories m_1 and m_2 such that $m_1\, \Psi\, m_2$, we have that $([\![\mathsf{G}_1]\!]\, m_1)\, \mathcal{L}(\Phi)\, ([\![\mathsf{G}_2]\!]\, m_2)$, where $\mathcal{L}(\Phi)$ denotes the lifting of Φ to distributions. Relational Hoare logic can be used to prove claims about the probability of events in games by using, for instance, the following rule:

$$\frac{m_1\, \Psi\, m_2 \qquad \models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi \qquad \Phi \implies (A\langle 1 \rangle \implies B\langle 2 \rangle)}{\Pr[\mathsf{G}_1, m_1 : A] \leq \Pr[\mathsf{G}_2, m_2 : B]}$$

Observational equivalence is defined by specializing the judgments to relations Ψ and Φ corresponding to the equality relation on subsets of program variables. Formally, let X be a set of variables, $m_1, m_2 \in \mathcal{M}$ and $f_1, f_2 : \mathcal{M} \to [0, 1]$. We define

$$\begin{aligned} m_1 =_X m_2 &\stackrel{\text{def}}{=} \forall x \in X.\, m_1(x) = m_2(x) \\ f_1 =_X f_2 &\stackrel{\text{def}}{=} \forall m_1\, m_2.\, m_1 =_X m_2 \implies f_1(m_1) = f_2(m_2) \end{aligned}$$

Then, two games G_1 and G_2 are observationally equivalent w.r.t. an input set of variables I and an output set of variables O, written $\models \mathsf{G}_1 \simeq^I_O \mathsf{G}_2$, iff $\models \mathsf{G}_1 \sim \mathsf{G}_2 : =_I \Rightarrow =_O$. Equivalently, $\models \mathsf{G}_1 \simeq^I_O \mathsf{G}_2$ iff for all memories $m_1, m_2 \in \mathcal{M}$ and functions $f_1, f_2 : \mathcal{M} \to [0, 1]$,

$$m_1 =_I m_2 \wedge f_1 =_O f_2 \implies [\![\mathsf{G}_1]\!]\, m_1\, f_1 = [\![\mathsf{G}_2]\!]\, m_2\, f_2$$

Observational equivalence is amenable to automation. CertiCrypt provides mechanized tactics based on dependency analyses to perform common program transformations

and to prove that two programs are observationally equivalent (note that observational equivalence is only a partial equivalence relation). The mechanized transformations include dead code elimination, call inlining, inter- and intra-procedural code motion and expression propagation.

We sometimes use a standard Hoare logic for reasoning about single programs. Its judgments are of the form $\{P\}\ G\ \{Q\}$, where G is a game and P and Q are predicates on states. Formally, a judgment $\{P\}\ G\ \{Q\}$ is valid iff for every memory $m \in \mathcal{M}$ and function $f : \mathcal{M} \to [0,1]$,

$$P\ m \wedge (\forall m.\ Q\ m \implies f(m) = 0) \implies [\![G]\!]\ m\ f = 0$$

This logic is subsumed by the relational Hoare logic,

$$\models \{P\}\ G\ \{Q\} \iff \models G \sim \mathsf{skip} : P\langle 1\rangle \Rightarrow Q\langle 1\rangle.$$

3 Statistical Distance

Statistical distance quantifies the largest difference between the probability that two distributions assign to the same event. We refer to [28] for an in-depth presentation of statistical distance and its properties. Formally, the statistical distance $\Delta(\mu_1, \mu_2)$ between two distributions μ_1 and μ_2 over a set A is defined as:

$$\Delta(\mu_1, \mu_2) \stackrel{\text{def}}{=} \sup_{f : A \to [0,1]} |\mu_1\ f - \mu_2\ f|$$

One important property of statistical distance that we frequently use in proofs is its invariance under function application, i.e. for any function $F : \mathcal{D}(A) \to \mathcal{D}(B)$ and distributions μ_1, μ_2 over A, $\Delta(F(\mu_1), F(\mu_2)) \leq \Delta(\mu_1, \mu_2)$.

Remark. In the traditional definition of statistical distance, f ranges only over Boolean-valued functions. Our definition is more convenient for reasoning about our monadic formalization of distributions. We have proved in Coq that the two definitions coincide for discrete distributions.

3.1 A Logic for Bounding Statistical Distance

Statistical distance admits a natural extension to programs; we define the statistical distance between two programs G_1 and G_2 as follows:

$$\Delta(\mathsf{G}_1, \mathsf{G}_2) \stackrel{\text{def}}{=} \max_{m,f} |[\![\mathsf{G}_1]\!]\ m\ f - [\![\mathsf{G}_2]\!]\ m\ f|$$

Or, fixing an initial memory m,

$$\Delta_m(\mathsf{G}_1, \mathsf{G}_2) \stackrel{\text{def}}{=} \max_{f} |[\![\mathsf{G}_1]\!]\ m\ f - [\![\mathsf{G}_2]\!]\ m\ f|$$

We define a logic that allows to upper bound $\Delta_m(\mathsf{G}_1, \mathsf{G}_2)$ by a function of the memory m; the logic deals with judgments of the form $(\!|\mathsf{G}_1, \mathsf{G}_2|\!) \preceq g$, where

$$(\!|\mathsf{G}_1, \mathsf{G}_2|\!) \preceq g \stackrel{\text{def}}{=} \forall m.\ \Delta_m(\mathsf{G}_1, \mathsf{G}_2) \leq g\ m \equiv \forall m\ f.\ |[\![\mathsf{G}_1]\!]\ m\ f - [\![\mathsf{G}_2]\!]\ m\ f| \leq g\ m$$

$$\frac{}{(\!|\mathsf{skip},\mathsf{skip}|\!) \preceq \lambda m.\, 0}[\text{Skip}] \qquad \frac{}{(\!|x \leftarrow e, x \leftarrow e|\!) \preceq \lambda m.\, 0}[\text{Ass}]$$

$$\frac{\forall m.\ \Delta\left([\![\mu_1]\!]\ m, [\![\mu_2]\!]\ m\right) \leq g\ m}{(\!|x \xleftarrow{\$} \mu_1, x \xleftarrow{\$} \mu_2|\!) \preceq g}[\text{Rnd}] \qquad \frac{(\!|c_1, c_2|\!) \preceq g \qquad (\!|c_1', c_2'|\!) \preceq g'}{(\!|c_1; c_1', c_2; c_2'|\!) \preceq \lambda m. [\![c_1]\!] m\ g' + g\ m}[\text{Seq}]$$

$$\frac{(\!|c_1, c_1'|\!) \preceq g_1 \qquad (\!|c_2, c_2'|\!) \preceq g_2}{(\!|\mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \mathsf{if}\ b\ \mathsf{then}\ c_1'\ \mathsf{else}\ c_2'|\!) \preceq \lambda m.\ \mathbf{if}\ [\![b]\!]\ m\ \mathbf{then}\ g_1\ m\ \mathbf{else}\ g_2\ m}[\text{Cond}]$$

$$\frac{(\!|c_1, c_2|\!) \preceq g \quad g_0(m) = 0 \quad g_{n+1}(m) = \mathbf{if}\ [\![b]\!]\ m\ \mathbf{then}\ [\![c_1]\!]\ m\ g_n + g(m)\ \mathbf{else}\ 0}{(\!|\mathsf{while}\ b\ \mathsf{do}\ c_1, \mathsf{while}\ b\ \mathsf{do}\ c_2|\!) \preceq \sup(\lambda n.\ g_n)}[\text{Whl}]$$

$$\frac{(\!|E_1(p), E_2(p)|\!) \preceq g \quad g =_X g \quad \forall x.\ x \in X \Rightarrow \mathsf{global}(x)}{(\!|y \leftarrow p(x), y \leftarrow p(x)|\!) \preceq g}[\text{Call}]$$

Fig. 1. Logic to bound the statistical distance between two probabilistic programs

Figure 1 presents the main rules of the logic; for readability, rules are stated for pairs of commands rather than pairs of programs, and assume that this pair of programs are executed in two fixed environments E_1 and E_2 respectively.

To prove the soundness, for instance, of the rule for sequential composition, we introduce an intermediate program $c_1; c_2'$ (where c_1 is executed in environment E_1 and c_2' in environment E_2) and prove that the distance between $[\![c_1; c_2']\!]\ m$ and $[\![c_1; c_1']\!]\ m$ is bounded by $[\![c_1]\!]\ m\ g'$, while the distance between $[\![c_1; c_2']\!]\ m$ and $[\![c_2; c_2']\!]\ m$ is bounded by $g\ m$. The rule for loops relies on the characterization of the semantics of a while loop as the least upper bound of its n-th unrolling $[\mathsf{while}\ e\ \mathsf{do}\ c]_n$, and on the auxiliary rule

$$\frac{(\!|[\mathsf{while}\ b\ \mathsf{do}\ c_1]_n, [\mathsf{while}\ b\ \mathsf{do}\ c_2]_n|\!) \preceq g_n}{(\!|\mathsf{while}\ b\ \mathsf{do}\ c_1, \mathsf{while}\ b\ \mathsf{do}\ c_2|\!) \preceq \sup(\lambda n.\ g_n)}$$

While the rules in Figure 1 are sufficient to reason about closed programs, they do not allow to reason about games in the presence of adversaries. We enhance the logic with a rule that allows to bound the statistical distance between calls to an adversary $\mathcal{A}$ executed in two different environments E_1 and E_2, i.e. it allows to draw conclusions of the form $(\!|\mathcal{A}, \mathcal{A}|\!) \preceq g$.[1] In its simplest formulation, the rule assumes that oracles are instrumented with a counter that keeps track of the number of queries made, and that the statistical distance between the distributions induced by a call to an oracle $x \leftarrow \mathcal{O}(\vec{e})$ in E_1 and E_2 is upper bounded by a constant ϵ, i.e. $(\!|\mathcal{O}, \mathcal{O}|\!) \preceq \epsilon$. In this case, the statistical distance between calls to the adversary $\mathcal{A}$ in E_1 and E_2 is upper bounded by $q \cdot \epsilon$, where q is an upper bound on the number of oracle calls made by the adversary.

For the application presented in Section 5, we need to formalize a more powerful rule, in which the statistical distance between two oracle calls can depend on the program state. Moreover, we allow the counter to be any integer expression, and only require that it does not decrease across oracle calls.

[1] For the sake of readability, we write $(\!|\mathcal{A}, \mathcal{A}|\!) \preceq g$ instead of $(\!|x \leftarrow \mathcal{A}(\vec{e}), x \leftarrow \mathcal{A}(\vec{e})|\!) \preceq g$, and likewise for oracles.

Lemma 1 (Adversary rule). *Let $\mathcal{A}$ be an adversary and let* cntr *be an integer expression whose variables cannot be written by $\mathcal{A}$. Let $h : \mathbb{N} \to [0,1]$ and define*

$$\bar{h}_{\mathsf{cntr}}(m, m') \stackrel{def}{=} \min\left(1, \sum_{i=[\![\mathsf{cntr}]\!]m}^{[\![\mathsf{cntr}]\!]m'-1} h(i)\right)$$

Assume that for every oracle $\mathcal{O}$,

$$(\!|\mathcal{O}, \mathcal{O}|\!) \preceq \lambda m.\ [\![E_1(\mathcal{O})]\!]\ m\ (\lambda m'.\ \bar{h}_{\mathsf{cntr}}(m, m'))$$

and $\{\mathsf{cntr} = i\}\ E_1(\mathcal{O})\ \{i \leq \mathsf{cntr}\}$. *Then,*

$$(\!|\mathcal{A}, \mathcal{A}|\!) \preceq \lambda m.\ [\![E_1(\mathcal{A})]\!]\ m\ \left(\lambda m'.\ \bar{h}_{\mathsf{cntr}}(m, m')\right)$$

3.2 Reasoning about Failure Events

Transitions based on failure events allow to transform a game into another game that is semantically equivalent unless some *failure* condition is triggered. The main tool to justify such transitions is the following lemma.

Lemma 2 (Fundamental Lemma). *Consider two games G_1, G_2 and let A, B, and F be events. If* $\Pr[\mathsf{G}_1 : A \wedge \neg F] = \Pr[\mathsf{G}_2 : B \wedge \neg F]$, *then*

$$|\Pr[\mathsf{G}_1 : A] - \Pr[\mathsf{G}_2 : B]| \leq \max\{\Pr[\mathsf{G}_1 : F], \Pr[\mathsf{G}_2 : F]\}$$

Note also that if, for instance, game G_2 is lossless, then $\Pr[\mathsf{G}_1 : F] \leq \Pr[\mathsf{G}_2 : F]$.

When $A = B$ and $F = \mathbf{bad}$ for some Boolean variable $\mathbf{bad}$, the hypothesis of the lemma can be automatically established by inspecting the code of both games: it holds if their code differs only after program points setting $\mathbf{bad}$ to true and $\mathbf{bad}$ is never reset to false. As a corollary, if two games G_1, G_2 satisfy this syntactic criterion and e.g. G_2 is lossless, $(\!|\mathsf{G}_1, \mathsf{G}_2|\!) \preceq \lambda m.\ \Pr[\mathsf{G}_2, m : \mathbf{bad}]$.

4 Weak Equivalences

In this section we introduce quantitative notions of program equivalence and equational theories to reason about them.

4.1 Approximate Observational Equivalence

Approximate observational equivalence generalizes observational equivalence between two games by allowing that their output distributions differ up to some quantity ϵ. Informally, two games G_1 and G_2 are ϵ-observationally equivalent w.r.t. an input set of variables I and an output set of variables O iff for every pair of memories m_1, m_2 coinciding on I,

$$\Delta([\![\mathsf{G}_1]\!]\ m_1)/=_O, ([\![\mathsf{G}_2]\!]m_2)/=_O) \leq \epsilon,$$

where for a distribution μ over a set A and an equivalence relation R on A, we let μ/R denote the quotient distribution of μ over A/R. For the purpose of formalization, it is more convenient to rely on the following alternative characterization that does not use quotient distributions, in part because the underlying language of Coq does not support quotient types.

Definition 1. *Two games* G_1 *and* G_2 *are* ϵ*-observationally equivalent w.r.t. an input set of variables* I *and an output set of variables* O*, written* $\models \mathsf{G}_1 \simeq^I_O \mathsf{G}_2 \preceq \epsilon$*, iff for all memories* $m_1, m_2 \in \mathcal{M}$ *and functions* $f_1, f_2 : \mathcal{M} \to [0,1]$

$$m_1 =_I m_2 \wedge f_1 =_O f_2 \implies |[\![\mathsf{G}_1]\!]\, m_1\, f_1 - [\![\mathsf{G}_2]\!]\, m_2\, f_2| \leq \epsilon$$

Figure 2 provides an excerpt of an equational theory for approximate observational equivalence; further and more general rules appear in the formal development. Most rules generalize observational equivalence in the expected way. For instance, the rule for random assignment considers the case of uniformly sampling over two finite sets A and B: in case $A = B$, one obtains $\epsilon = 0$.

$$\frac{\models c_1 \simeq^I_O c_2 \preceq \epsilon_1 \qquad \models c_2 \simeq^I_O c_3 \preceq \epsilon_2}{\models c_1 \simeq^I_O c_3 \preceq \epsilon_1 + \epsilon_2} \qquad \frac{\models c_1 \simeq^{I'}_{O'} c_2 \preceq \epsilon' \quad I' \subseteq I \quad O \subseteq O' \quad \epsilon' \leq \epsilon}{\models c_1 \simeq^I_O c_2 \preceq \epsilon}$$

$$\frac{\models c_1 \simeq^I_{O'} c_2 \preceq \epsilon_1 \quad \models c_1' \simeq^{O'}_O c_2' \preceq \epsilon_2}{\models c_1; c_1' \simeq^I_O c_2; c_2' \preceq \epsilon_1 + \epsilon_2}$$

$$\frac{\models c_1 \simeq^I_O c_1' \preceq \epsilon \quad \models c_2 \simeq^I_O c_2' \preceq \epsilon \qquad \forall m, m'.\ I\ m\ m' \implies [\![b]\!]\, m = [\![b']\!]\, m'}{\models \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \simeq^I_O \mathsf{if}\ b'\ \mathsf{then}\ c_1'\ \mathsf{else}\ c_2' \preceq \epsilon}$$

$$\frac{\epsilon = \#(A \cap B)|\tfrac{1}{\#A} - \tfrac{1}{\#B}| + \max\left\{\tfrac{\#(A \setminus B)}{\#A}, \tfrac{\#(B \setminus A)}{\#B}\right\}}{\models x \xleftarrow{\$} A \simeq^I_{I \cup \{x\}} x \xleftarrow{\$} B \preceq \epsilon}$$

Fig. 2. Selected rules for reasoning about approximate observational equivalence

4.2 A Conditional Variant

The application we describe in Section 5 requires reasoning about conditional approximate observational equivalence, a generalization of approximate observational equivalence. We define for each distribution μ and event P the conditional distribution $\mu\,|_P$ as

$$\mu\,|_P \stackrel{\text{def}}{=} \lambda f.\ \mu\left(\lambda a.\ \frac{f(a)\ \mathbb{1}_P(a)}{\mu\ \mathbb{1}_P}\right)$$

Intuitively, $\mu\,|_P\ \mathbb{1}_Q$ yields the conditional probability of Q given P.

Definition 2. *A game* G_1 *conditioned on predicate* P_1 *is* ϵ*-observationally equivalent to a game* G_2 *conditioned on* P_2 *w.r.t. an input set of variables* I *and an output set of variables* O*, written* $\models [\mathsf{G}_1]_{P_1} \simeq_O^I [\mathsf{G}_2]_{P_2} \preceq \epsilon$*, iff for any* $m_1, m_2 \in \mathcal{M}$ *and* $f_1, f_2 : \mathcal{M} \to [0,1]$*,*

$$m_1 =_I m_2 \wedge f_1 =_O f_2 \implies |([\![\mathsf{G}_1]\!]\, m_1)\,|_{P_1}\ f_1 - ([\![\mathsf{G}_2]\!]\, m_2)\,|_{P_2}\ f_2| \leq \epsilon$$

Conditional approximate observational equivalence subsumes classic approximate observational equivalence, which can be recovered by taking $P_1 = P_2 = \mathsf{true}$.

5 Indifferentiability

In this section we present an application of the techniques introduced above to prove the security of cryptographic constructions in the indifferentiability framework of Maurer et al. [23]. In particular, we consider the notion of indifferentiability from a random oracle. A random oracle is an ideal primitive that maps elements in some domain into uniformly and independently distributed values in a finite set; queries are answered consistently so that identical queries are given the same answer. A proof conducted in the random oracle model for a function $h : A \to B$ assumes that h is made publicly available to all parties.

Definition 3 (Indifferentiability). *A procedure* $\mathcal{F}$ *that has access to a random oracle* $h : \{0,1\}^* \to A$ *is said to be* $(t_{\mathcal{S}}, t_{\mathcal{D}}, q_1, q_2, \epsilon)$*-indifferentiable from a random oracle* $\mathcal{H} : \{0,1\}^* \to B$ *if there exists a simulator* $\mathcal{S}$ *with oracle access to* $\mathcal{H}$ *and executing within time* $t_{\mathcal{S}}$ *such that any distinguisher* $\mathcal{D}$ *running within time* $t_{\mathcal{D}}$ *and making at most* q_1 *queries to an oracle* $\mathcal{O}_1$ *and* q_2 *queries to an oracle* $\mathcal{O}_2$ *has at most probability* ϵ *of distinguishing a scenario where* $\mathcal{O}_1$ *is implemented as* $\mathcal{F}$ *and* $\mathcal{O}_2$ *as* h *from a scenario where* $\mathcal{O}_1$ *is implemented as* H *and* $\mathcal{O}_2$ *as* $\mathcal{S}$ *instead. Put in terms of games,*

Game $\mathsf{G} : \mathbf{L} \leftarrow \mathsf{nil};\ b \leftarrow \mathcal{D}()$
Oracle $\mathcal{O}_1(x)$: return $\mathcal{F}(x)$
Oracle $\mathcal{O}_2(x)$:
if $x \notin \mathsf{dom}(\mathbf{L})$ then
$\quad y \xleftarrow{\$} A;\ \mathbf{L}(x) \leftarrow y$
return $\mathbf{L}(x)$

Game $\mathsf{G}' : \mathbf{L} \leftarrow \mathsf{nil};\ b \leftarrow \mathcal{D}()$
Oracle $\mathcal{O}_1(x)$:
if $x \notin \mathsf{dom}(\mathbf{L})$ then
$\quad y \xleftarrow{\$} B;\ \mathbf{L}(x) \leftarrow y$
return $\mathbf{L}(x)$
Oracle $\mathcal{O}_2(x)$: return $\mathcal{S}(x)$

$$|\Pr[\mathsf{G} : b = \mathsf{true}] - \Pr[\mathsf{G}' : b = \mathsf{true}]| \leq \epsilon$$

Random oracles into elliptic curves over finite fields are typically built from a random oracle h on the underlying field and a deterministic encoding f that maps elements of the field into the elliptic curve. Examples of such encodings include Icart function [21] and the Shallue-Woestijne-Ulas (SWU) algorithm [27]. In general, and in particular for the aforementioned mappings, the function f is not surjective and only covers a fraction of points in the curve. Hence, the naive definition of a hash function H as $f \circ h$ would not cover the whole curve, contradicting the assumption that H behaves as a random oracle. In a recent paper, Brier et al. [9] show how to build hash functions into

elliptic curves that are indifferentiable from a random oracle from a particular class of encodings, including both SWU and Icart encodings.

We prove the indifferentiability of the construction put forward by Brier et al. in the formal framework of CertiCrypt. The proof introduces two intermediate constructions and is structured in three steps:

1. We first prove that any efficiently invertible encoding f can be turned into a *weak encoding* (Theorem 1);
2. We then show an efficient construction to transform any weak encoding f into an *admissible encoding* (Theorem 2);
3. Finally, we prove that any admissible encoding can be turned into a hash function indifferentiable from a random oracle (Theorem 3).

Moreover, we show in Sect. 6 that Icart encoding is efficiently invertible and thus yields a hash function indifferentiable from a random oracle when plugged in into the above construction. We recall the alternative definitions of weak and admissible encoding from [22]. Note that these do not match the definitions in [9], but, in comparison, are better behaved: e.g. admissible encodings as we define them are closed under functional composition and cartesian product.

Definition 4 (Weak encoding). *A function $f : S \to R$ is an (α, ϵ)-weak encoding if it is computable in polynomial-time and there exists a probabilistic polynomial-time algorithm $\mathcal{I}_f : R \to S_\perp$ such that*

1. $\{\mathsf{true}\}\ r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r)\ \{s = \perp \vee f(s) = r\}$
2. $\models [r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r)]_{s \neq \perp} \simeq^{\emptyset}_{\{s\}} [s \xleftarrow{\$} S] \preceq \epsilon$
3. $\Pr[r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) : s = \perp] \leq 1 - \alpha^{-1}$

Definition 5 (Admissible encoding). *A function $f : S \to R$ is an ϵ-admissible encoding if it is computable in polynomial-time and there exists a probabilistic polynomial-time algorithm $\mathcal{I}_f : R \to S_\perp$ such that*

1. $\{\mathsf{true}\}\ r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r)\ \{s = \perp \vee f(s) = r\}$
2. $\models r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) \simeq^{\emptyset}_{\{s\}} s \xleftarrow{\$} S \preceq \epsilon$

Brier et al. [9] prove that if $\mathbb{G}$ is a finite cyclic group of order N with generator g, a function into $\mathbb{G}$ indifferentiable from a random oracle can be built from any polynomially invertible function $f : A \to \mathbb{G}$ and hash functions $h_1 : \{0,1\}^\star \to A$ and $h_2 : \{0,1\}^* \to \mathbb{Z}_N$ as follows:

$$H(m) \stackrel{\text{def}}{=} f(h_1(m)) \otimes g^{h_2(m)} \tag{1}$$

Intuitively, the term $g^{h_2(m)}$ behaves as a one-time pad and ensures that H covers all points in the group even if f covers only a fraction. Our proof generalizes this construction to finitely generated abelian groups.

We begin by showing that any efficiently invertible encoding is a weak encoding.

Theorem 1. *Let $f : S \to R$ be a function computable in polynomial-time such that for any $r \in R$, $\#f^{-1}(r) \leq B$. Assume there exists a polynomial-time algorithm $\mathcal{I}$ that given $r \in R$ outputs the set $f^{-1}(r)$. Then, f is an $(\alpha, 0)$-weak encoding, with $\alpha = B\ \#R/\#S$.*

Proof. Using $\mathcal{I}$, we build a partial inverter $\mathcal{I}_f : R \to S_\perp$ of f that satisfies the properties in Definition 4:

$$\begin{array}{l}\mathcal{I}_f(r) : X \leftarrow \mathcal{I}(r);\ b \xleftarrow{\$} \mathsf{true} \oplus_{\#X/B} \mathsf{false};\\ \qquad\quad \mathsf{if}\ b = \mathsf{true}\ \mathsf{then}\ s \xleftarrow{\$} X;\ \mathsf{return}\ s\ \mathsf{else}\ \mathsf{return}\ \perp\end{array}$$

First observe that $\mathcal{I}_f(r)$ fails with probability $1-\#f^{-1}(r)/B$ or else returns an element uniformly chosen from the set of pre-images of r, and thus satisfies the first property trivially. In addition, for any $x \in S$ we have

$$\begin{aligned}\Pr\left[r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) : s = x\right] &= \frac{1}{B\#R}\\ \Pr\left[r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) : s \neq \perp\right] &= \frac{1}{\#R}\sum_{r\in R}\frac{\#f^{-1}(r)}{B} = \frac{\#S}{B\#R}\end{aligned}$$

Hence, for a uniformly chosen r, the probability of $\mathcal{I}_f(r)$ failing is exactly $1-\alpha^{-1}$, and the probability of returning any particular value in S conditioned to not failing is uniform. □

We show next how to construct an admissible encoding from a weak-encoding into a finite abelian group. Recall that every finite abelian group $\mathbb{G}$ is isomorphic to a product of cyclic groups[2]

$$\mathbb{G} \simeq \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$$

If we fix generators g_i for each $\mathbb{Z}_{n_i}$, then any $x \in \mathbb{G}$ admits a unique representation as a vector $(g_1^{z_1}, \ldots, g_k^{z_k})$. We use $\log$ to denote the operator that returns the canonical representation $\vec{z} = (z_1, \ldots, z_k)$ for any $x \in \mathbb{G}$.

Theorem 2. *Let $\mathbb{G} \simeq \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$ be a finite abelian group and let g_i be a generator of $\mathbb{Z}_{n_i}$ for $i = 1 \ldots k$. Assume that $f : A \to \mathbb{G}$ is an (α, ϵ)-weak encoding. Then, for any polynomially bounded T, the function*

$$\begin{aligned}F &: A \times \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k} \to \mathbb{G}\\ F(a, z_1, \ldots, z_k) &= f(a) \otimes g_1^{z_1} \otimes \cdots \otimes g_k^{z_k}\end{aligned}$$

is an ϵ'-admissible encoding into $\mathbb{G}$, with $\epsilon' = \epsilon + \left(1-\alpha^{-1}\right)^{T+1}$.

Proof. Since f is a weak encoding, there exists a polynomial-time computable inverter $\mathcal{I}_f$ of f satisfying the conditions in Definition 4. Let $T \in \mathbb{N}$ be polynomially bounded. Using $\mathcal{I}_f$, we build a partial inverter $\mathcal{I}_F$ of F that satisfies the properties in Definition 5:

$$\begin{array}{l}\mathcal{I}_F(r) : i \leftarrow 0;\ a \leftarrow \perp;\\ \qquad \mathsf{while}\ (i \leq T \wedge a = \perp)\ \mathsf{do}\\ \qquad\quad \vec{z} \xleftarrow{\$} \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k};\\ \qquad\quad x \leftarrow r \otimes g_1^{-z_1} \otimes \cdots \otimes g_k^{-z_k};\\ \qquad\quad a \leftarrow \mathcal{I}_f(x);\ i \leftarrow i+1\\ \qquad \mathsf{end};\\ \qquad \mathsf{if}\ a \neq \perp\ \mathsf{then}\ \mathsf{return}\ (a, \vec{z})\ \mathsf{else}\ \mathsf{return}\ \perp\end{array}$$

[2] The decomposition can be made unique by fixing additional conditions on $n_1 \ldots n_k$.

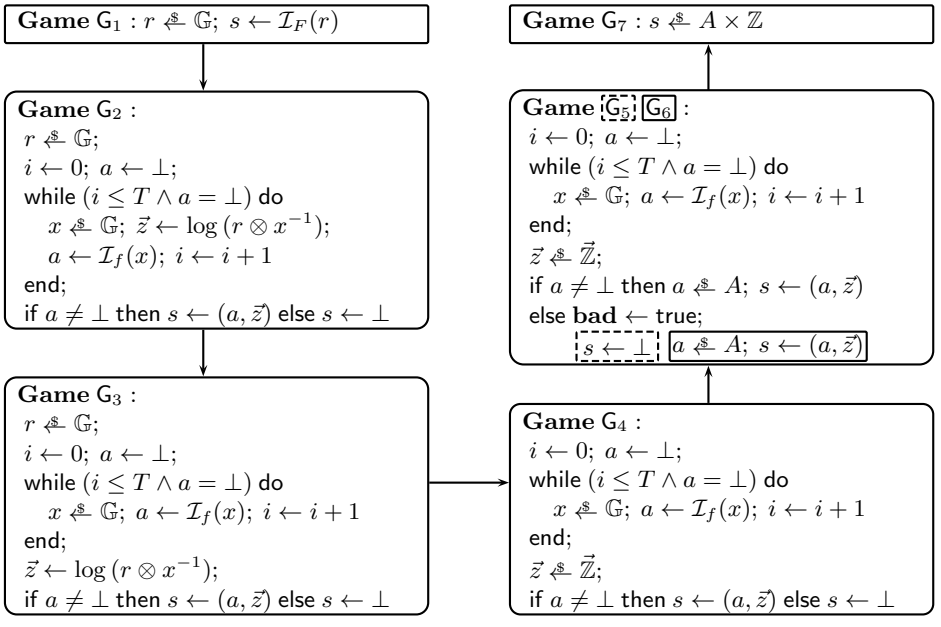

Fig. 3. Sequence of games used in Theorem 2

The partial inverter $\mathcal{I}_F$ runs in time $t_{\mathcal{I}_F} = (T+1)\ t_{\mathcal{I}_f}$, where $t_{\mathcal{I}_f}$ is a bound on the running time of $\mathcal{I}_f$. Hence, $\mathcal{I}_F$ is polynomial-time for any polynomially bounded T.

For the sake of readability in the following we use $\vec{\mathbb{Z}}$ to denote $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$ and $\vec{g}^{\vec{z}}$ to denote $g_1^{z_1} \otimes \cdots \otimes g_k^{z_k}$. We prove that

$$\models r \xleftarrow{\$} \mathbb{G};\ s \leftarrow \mathcal{I}_F(r) \simeq^{\emptyset}_{\{s\}} s \xleftarrow{\$} A \times \vec{\mathbb{Z}} \preceq \epsilon'$$

using the sequence of games $\mathsf{G}_1, \ldots, \mathsf{G}_7$ shown in Figure 3, the mechanized program transformations of CertiCrypt, and the proof rules for observational and approximate observational equivalence. We briefly describe the proof below.

We obtain game G_2 by first inlining the call to $\mathcal{I}_F$ in the initial game and then applying the following algebraic equivalence to transform the body of the while loop:

$$\models \vec{z} \xleftarrow{\$} \vec{\mathbb{Z}};\ x \leftarrow r \otimes \vec{g}^{-\vec{z}} \simeq^{\{r\}}_{\{r,x,z\}} x \xleftarrow{\$} \mathbb{G};\ \vec{z} \leftarrow \log(r \otimes x^{-1})$$

We obtain game G_3 by moving the assignment to $\vec{z}$ outside the loop in game G_2. This transformation is semantics-preserving because $\vec{z}$ is never used inside the loop and the value that it has when exiting the loop only depends on the value of x in the last iteration. Formally, this is proven by unfolding the first iteration of the loop and establishing that the relation

$$=_{\{i,x,a,r\}} \wedge (\vec{z} = \log(r \otimes x^{-1}))\langle 1 \rangle$$

is a relational invariant between the loop in G_2 and the loop resulting from removing the assignment to $\vec{z}$. By appending $\vec{z} \leftarrow \log(r \otimes x^{-1})$ to the latter loop, we recover equivalence on $\vec{z}$.

Since r is no longer used inside the loop, we can postpone its definition after the loop, and use the following algebraic equivalence to sample $\vec{z}$ instead of r:

$$\models r \xleftarrow{\$} \mathbb{G};\ \vec{z} \leftarrow \log\left(r \otimes x^{-1}\right) \simeq^{\{x\}}_{\{r,x,z\}} \vec{z} \xleftarrow{\$} \vec{\mathbb{Z}};\ r \leftarrow x \otimes \vec{g}^{\vec{z}}$$

We obtain G_4 by additionally removing the assignment to r, which is now dead code.

For the next step in the proof we use the fact that f is a weak encoding and therefore the distribution of a after a call $a \leftarrow \mathcal{I}_f(x)$ conditioned to $a \neq \perp$ is ϵ-away from the uniform distribution. This allows us to resample the value of a after the loop, provided $a \neq \perp$, incurring a penalty ϵ on the statistical distance of the distribution of s between G_4 and G_5. To prove this formally, let b be the condition of the loop and c its body. Observe that the semantics of the loop coincides with the semantics of its $(T+1)$-unrolling $[\mathsf{while}\ b\ \mathsf{do}\ c]_{T+1}$. We show by induction on T that for any $[0,1]$-valued functions f, g s.t. $f =_{\{a'\}} g$,

$$m_1 =_{\{a,i\}} m_2 \ \wedge\ m_1(a) = \perp \implies |[\![c_1]\!]\, m_1\, f' - [\![c_2]\!]\, m_2\, g'| \leq \epsilon$$

where

$$\begin{aligned} c_1 &= [\mathsf{while}\ b\ \mathsf{do}\ c]_{T+1};\ \mathsf{if}\ a \neq \perp\ \mathsf{then}\ a' \leftarrow a \\ c_2 &= [\mathsf{while}\ b\ \mathsf{do}\ c]_{T+1};\ \mathsf{if}\ a \neq \perp\ \mathsf{then}\ a' \xleftarrow{\$} A \\ f'(m) &= \mathbf{if}\ m(a) \neq \perp\ \mathbf{then}\ f(m)\ \mathbf{else}\ 0 \\ g'(m) &= \mathbf{if}\ m(a) \neq \perp\ \mathbf{then}\ g(m)\ \mathbf{else}\ 0 \end{aligned}$$

and use this to conclude the ϵ-approximate equivalence of G_4 and G_5.

Since G_5 and G_6 are syntactically equivalent except for code appearing after the flag $\mathbf{bad}$ is set, we apply the corollary of the Fundamental Lemma in Section 3.2 to obtain the bound

$$(\!|\mathsf{G}_5, \mathsf{G}_6|\!) \preceq \Pr\left[\mathsf{G}_5 : \mathbf{bad}\right]$$

Since the probability of failure of $\mathcal{I}_f$ on a uniformly chosen input is upper bounded by $1 - \alpha^{-1}$, we can show by induction on T that

$$\Pr\left[\mathsf{G}_5 : \mathbf{bad}\right] \leq \left(1 - \alpha^{-1}\right)^{T+1},$$

from which we conclude $\models \mathsf{G}_5 \simeq^{\emptyset}_{\{s\}} \mathsf{G}_6 \preceq \left(1 - \alpha^{-1}\right)^{T+1}$.

By coalescing the branches in the conditional at the end of G_6 and removing dead code, we prove that the game is observational equivalent w.r.t a and $\vec{z}$ to the game $a \xleftarrow{\$} A;\ \vec{z} \xleftarrow{\$} \vec{\mathbb{Z}};\ s \leftarrow (a, z)$, which is trivially equivalent to G_7.

By composing the above results, we conclude

$$\models \mathsf{G}_1 \simeq^{\emptyset}_{\{s\}} \mathsf{G}_7 \preceq \epsilon + \left(1 - \alpha^{-1}\right)^{T+1} \tag{2}$$

We must also show that $s = \perp\ \vee F(s) = r$ is a post-condition of G_1. As G_1 and G_3 are observationally equivalent with respect to s and r, it is sufficient to establish the validity of the post-condition for G_3. We show that $a \neq \perp \Rightarrow x = f(a)$ is an invariant of the loop. When the loop finishes, either $a = \perp$ and in this case $s = \perp$, or $a \neq \perp$ and we have $F(s) = f(a) \otimes \vec{g}^{\vec{z}} = x \otimes r \otimes x^{-1} = r$. □

Finally, we show that the composition of an admissible encoding $f : S \to R$ and a random oracle into S is indifferentiable from a random oracle into R.

Theorem 3. *Let $f : S \to R$ be an ϵ-admissible encoding with inverter algorithm $\mathcal{I}_f$ and let $h : \{0,1\}^\star \to S$ be a random oracle. Then, $f \circ h$ is $(t_{\mathcal{S}}, t_{\mathcal{D}}, q_1, q_2, \epsilon')$-indifferentiable from a random oracle into R, where $t_{\mathcal{S}} = q_1\, t_{\mathcal{I}_f}$ and $\epsilon' = 2(q_1 + q_2)\epsilon$.*

Before moving to the proof of Theorem 3, we prove the following useful result.

Lemma 3. *Let $f : S \to R$ be an ϵ-admissible encoding with inverter algorithm $\mathcal{I}_f$. Then*

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

Proof. Define

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

Since the first branch of the conditional in c_1 is never executed, we have:

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

Due to the second property of Definition 5, the distributions of s after executing c_i and c_f are ϵ-away. Using the rules for approximate observational equivalence, we obtain

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

The corollary to the Fundamental Lemma in Section 3.2 implies that $(\!|c_2, c_3|\!) \preceq \Pr[c_2 : \mathbf{bad}]$. Moreover,

$$\Pr[c_2 : \mathbf{bad}] = 1 - \Pr[c_f : s \neq \bot] = \Pr[s \xleftarrow{\$} S : s \neq \bot] - \Pr[c_f : s \neq \bot] \leq \epsilon$$

where the last inequality holds again because of the second property of Definition 5. Since the final values of r and s in programs c_2 and c_3 are independent of the initial memory, we have

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

Because $\mathcal{I}_f$ is a partial inverter for f, the else branch of the conditional in c_3 has no effect and can be removed, and thus $\models c_3 \simeq^{\emptyset}_{\{r,s\}} c_f$. We conclude by transitivity of approximate observational equivalence. □

Proof (of Theorem 3). Let $\mathcal{D}$ be a distinguisher against the indifferentiability of $f \circ h$ making at most q_1 queries to $\mathcal{O}_1$ and at most q_2 queries to $\mathcal{O}_2$. We exhibit a simulator $\mathcal{S}$ that uses a random oracle into R to simulate h and show that $\mathcal{D}$ cannot distinguish a game G where $\mathcal{O}_1$ and $\mathcal{O}_2$ are implemented by $f \circ h$ and h respectively from a game G' where they are implemented by $\mathcal{S}$ and a random oracle into R instead. An overview

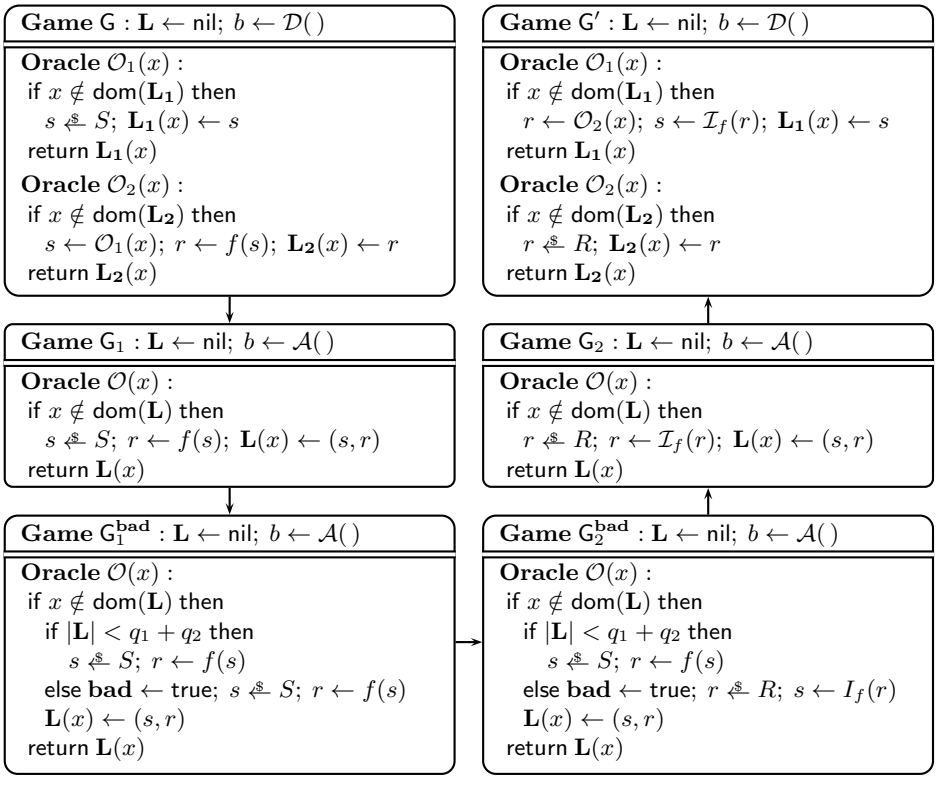

Fig. 4. Games used in the proof of Theorem 3

of the proof, including these two games and the definition of the simulator is shown in Figure 4.

Our goal is to prove

$$|\Pr[\mathsf{G} : b = \mathsf{true}] - \Pr[\mathsf{G}' : b = \mathsf{true}]| \leq 2(q_1 + q_2)\epsilon \tag{3}$$

The crux of the proof is an application of Lemma 1. In order to apply it, we need first to transform the initial games to replace oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ by a single joint oracle that simultaneously returns the responses of both. Using $\mathcal{D}$, we construct an adversary $\mathcal{A}$ with access to a single joint oracle, such that games G and G' are equivalent to games G_1 and G_2 in the figure. Adversary $\mathcal{A}$ simply calls the distinguisher $\mathcal{D}$ and forwards the value it returns; it simulates $\mathcal{O}_1$ and $\mathcal{O}_2$ by using its own oracle $\mathcal{O}$.

We assume without loss of generality the equivalence between games G and G_1, and G' and G_2, respectively. This is identical to the assumption in [9] that the distinguisher always makes the same queries to both its oracles. Games G_1 and G_2 satisfy the equalities:

$$\Pr[\mathsf{G} : b = \mathsf{true}] = \Pr[\mathsf{G}_1 : b = \mathsf{true}] \qquad \Pr[\mathsf{G}' : b = \mathsf{true}] = \Pr[\mathsf{G}_2 : b = \mathsf{true}]$$

Furthermore, since $\mathcal{D}$ makes at most q_1 queries to $\mathcal{O}_1$ and q_2 queries to oracle $\mathcal{O}_2$, $\mathcal{A}$ makes at most $q = q_1 + q_2$ queries to its oracle.

We next transform the implementation of oracle $\mathcal{O}$ in games G_1 and G_2 to enforce the bound $q_1 + q_2$ on the total number of queries. After the allotted number of queries is exhausted, oracle $\mathcal{O}$ behaves the same way in the two games. This ensures that further queries will not make the statistical distance between the two games grow and paves the way to applying Lemma 1. This transformation preserves observational equivalence because we know that $\mathcal{A}$ will not make more queries than allowed. One way of justifying this is using the syntactic criterion for Lemma 2: we annotate the games with a flag $\mathbf{bad}$ that is set to true at points where the implementations of the oracle $\mathcal{O}$ in the games differ and obtain

$$\Pr\left[\mathsf{G}_1^{\mathbf{bad}} : b = \mathsf{true} \wedge \neg\mathbf{bad}\right] = \Pr\left[\mathsf{G}_2^{\mathbf{bad}} : b = \mathsf{true} \wedge \neg\mathbf{bad}\right]$$

But since $\mathbf{bad} \implies q < |\mathbf{L}|$ is an invariant and $|\mathbf{L}| \leq q$ is a post-condition of both games,

$$\Pr\left[\mathsf{G}_1^{\mathbf{bad}} : b = \mathsf{true}\right] = \Pr\left[\mathsf{G}_2^{\mathbf{bad}} : b = \mathsf{true}\right]$$

We can now apply Lemma 1 to the games G_2 and $\mathsf{G}_2^{\mathbf{bad}}$, defining $\mathsf{cntr} = |\mathbf{L}|$ and $h(i) = \mathbf{if}\ i < q\ \mathbf{then}\ 2\epsilon\ \mathbf{else}\ 0$. The second hypothesis of the lemma, i.e that a call to $E_2(\mathcal{O})$ cannot decrease $|\mathbf{L}|$, is immediate. We can assume that $2q\epsilon < 1$ (otherwise the theorem is trivially true). Then,

$$\sum_{i=[\![\mathsf{cntr}]\!]m}^{[\![\mathsf{cntr}]\!]m'-1} h(i) \leq 2q\epsilon < 1, \qquad \text{and} \qquad \bar{h}_{\mathsf{cntr}}(m, m') = \sum_{i=[\![\mathsf{cntr}]\!]m}^{[\![\mathsf{cntr}]\!]m'-1} h(i)$$

We are only left to prove that

$$(\!|E_2(\mathcal{O}), E_2^{\mathbf{bad}}(\mathcal{O})|\!) \preceq \lambda m.\ [\![E_2(\mathcal{O})]\!]\ m\ (\lambda m'.\ \bar{h}_{\mathsf{cntr}}(m, m'))$$

Doing a case analysis on the conditions $x \in \mathsf{dom}(\mathbf{L})$ and $|\mathbf{L}| < q$ yields four cases; three of them yield a null distance and are immediate. The remaining case, where $x \notin \mathsf{dom}(\mathbf{L})$ and $|\mathbf{L}| < q$, yields a distance 2ϵ and follows from Lemma 3. We finally obtain $(\!|\mathsf{G}_2, \mathsf{G}_2^{\mathbf{bad}}|\!) \preceq 2(q_1+q_2)\epsilon$, which combined with the previous results implies the desired inequality. □

6 Application to Elliptic Curves

This section discuss the application of the proof presented in the previous section to hashing into elliptic curves.

Let $\mathbb{F}_{p^m}$ be a finite field of cardinal p^m, with $p > 3$ prime. An elliptic curve over $\mathbb{F}_{p^m}$ is defined by the equation $Y^2 = X^3 + aX + b$ where the parameters a, b are elements of $\mathbb{F}_{p^m}$ such that $4a^3 + 27b^2 \neq 0$ (the curve must be non-singular). The set of points of such a curve, which we denote $\mathbb{E}_{a,b}$, can be construed as a finite abelian group with the *point at infinite* O as the identity element. Furthermore, it can be shown that the group $\mathbb{E}_{a,b}$ is either cyclic or a product of two cyclic groups.

Hence, applying the results from the previous section, any polynomially invertible function into a $\mathbb{E}_{a,b}$ can be transformed into a hash function that is indifferentiable from a random oracle. In particular, this holds for Icart encoding, as we show next.

For $p^m \equiv 2 \pmod 3$, Icart function $f_{a,b} : \mathbb{F}_{p^m} \to \mathbb{E}_{a,b}$ is defined as:

$$f_{a,b}(u) \stackrel{\text{def}}{=} \begin{cases} (x, ux + v) & \text{if } u \neq 0 \\ ((-b)^{\frac{1}{3}}, 0) & \text{if } u = 0 \wedge a = 0 \\ O & \text{if } u = 0 \wedge a \neq 0 \end{cases} \tag{4}$$

$$\text{where} \qquad x = \left(v^2 - b - \frac{u^6}{27}\right)^{\frac{1}{3}} + \frac{u^2}{3} \qquad v = \frac{3a - u^4}{6u}$$

As a side remark, observe that the original definition only deals with the case $a \neq 0$; the definition for the case $a = 0$ was suggested to us by Thomas Icart in a private communication.

The set of pre-images of a point in the curve under Icart function can be computed efficiently by solving for the roots of polynomials over $\mathbb{F}_{p^m}$ of degree at most 4—any point in the curve has at most 4 pre-images:

$$f_{a,b}^{-1}(O) \stackrel{\text{def}}{=} \begin{cases} \{0\} & \text{if } a \neq 0 \\ \emptyset & \text{if } a = 0 \end{cases} \qquad f_{a,b}^{-1}(X, Y) \stackrel{\text{def}}{=} \begin{cases} \{u | u^3 - 6uX + 6Y = 0\} & \text{if } a = 0 \\ \{u | u^4 - 6u^2X + 6uY = 3a\} & \text{if } a \neq 0 \end{cases}$$

This can be done using any efficient algorithm for factoring polynomials over finite fields, e.g. Berlekamp's algorithm. Thus, Icart encoding is polynomially invertible.

Formalization. To apply our generic proof of indifferentiability to Icart function, we proceeded as follows:

1. We integrated Théry's formalization of elliptic curves [30] in our framework, and showed that the set of points of the elliptic curve $\mathbb{E}_{a,b}$ can be construed as a finite cyclic group, as defined in SSReflect standard library [19];
2. We defined Icart function, and showed that it generates points in the curve $\mathbb{E}_{a,b}$. This required showing the existence of cubic roots in the field $\mathbb{F}_{p^m}$ (the cubic root of $x \in \mathbb{F}_{p^m}$ is the element $x^{(2p^m-1)/3}$);
3. We defined the inverse of Icart function, for which we need to assume an efficient method for factoring polynomials of degree 4 over the underlying field, as no existing Coq library readily provides the necessary background;
4. We applied Theorem 1 to show that Icart function is an $(\alpha, 0)$-weak encoding, where $\alpha = 4N/p^m$ where N is the order of $\mathbb{E}_{a,b}$;
5. We applied Theorem 2 to show that for any polynomially bounded T, the function $F : \mathbb{F}_{p^m} \times \mathbb{Z}_N$, defined as $F(u, z) = f_{a,b}(u) + g^z$, where g is a generator of $\mathbb{Z}_N$, is an ϵ-admissible encoding, where $\epsilon = (1 - \alpha^{-1})^{T+1}$;
6. We finally applied Theorem 3 to show that if F is composed with a random oracle into $\mathbb{F}_{p^m} \times \mathbb{Z}_N$ (equivalently, a random oracle into $\mathbb{F}_{p^m}$ and a random oracle into $\mathbb{Z}_N$), the resulting construction is $(t_{\mathcal{S}}, t_{\mathcal{D}}, q_1, q_2, 2(q_1 + q_2)\epsilon)$-indifferentiable from a random oracle into $\mathbb{E}_{a,b}$, where $t_{\mathcal{S}} = q_1\ t_{\mathcal{I}_F} = q_1\ (T + 1)\ t_{f^{-1}}$ and $t_{f^{-1}}$ is an upper bound on the time needed to compute the pre-image of a point under Icart function, i.e. to solve a polynomial of degree 4 in $\mathbb{F}_{p^m}$.

7 Related Work

Weak Equivalences. The impossibility to achieve perfect security has motivated several proposals for weaker, quantitative, definitions of security. Prominent examples include notions of confidentiality based on information theory [11, 12, 24, 29]. More recently, Dwork [14] has suggested differential privacy as an alternative notion that quantifies the privacy guaranteed by confidential data analysis. All of these definitions can be construed as quantitative hyperproperties [12], and readily extend to relational properties that are closely related to statistical distance.

Approximate observational equivalence is also closely related to weak notions of bisimulations [26]. Barthe et al. [6] generalize approximate observational equivalence to an approximate relational Hoare logic and report on an extension of the CertiCrypt framework for reasoning about differential privacy. The validity of judgments in this logic is based on a notion of approximate lifting of a relation that is closely related to the notion used in [26].

Hashing into Elliptic Curves. A number of highly relevant cryptographic constructions, including identity based schemes [8] and password based key exchange protocols [7], require hashing into elliptic curves. Indeed, there have been a number of proposals for such hash functions, see for instance [17, 21, 27]. Recently, Farashahi et al. [15] developed powerful techniques to show the indifferentiability of hash function constructions based on deterministic encodings. Their results improve on [9], in the sense that they apply to a larger class of encodings, including encodings to hyperelliptic curves, and that they provide tighter bounds for encodings that are covered by both methods.

Formalization and Verification of Elliptic Curves. To our best knowledge, our work provides the first machine-checked proof of security for a cryptographic primitive based on elliptic curves. There are, however, previous works on the formalization of elliptic curves: Hurd, Gordon and Fox [20] report on the verification in HOL of the group laws and an application to the functional correctness of ElGamal encryption. Théry and Hanrot [30] use Coq to formalize the group laws, and show how the formalization of elliptic curves can be used to build efficient reflective tactics for testing primality.

8 Conclusion

This paper reports on a machine-checked proof of a recent construction to build hash functions that are indifferentiable from a random oracle into an elliptic curve. The example is singular among other examples that have been formalized using CertiCrypt, because it involves complex reasoning about algebraic geometry and requires the formalization of new weak forms of program equivalence.

The formalization establishes the ability of CertiCrypt to integrate smoothly with existing libraries of complex mathematics. Overall, the formalization consists of over 65,000 lines of Coq (without counting components reused from the standard libraries of Coq and SSReflect), which break down as follows: 45,000 lines corresponding to the original CertiCrypt framework, 3,500 lines of extensions to CertiCrypt, 7,000 lines

written originally for our application to indifferentiability, and 10,000 lines of a slightly adapted version of Théry [30] elliptic curve library.

Our work paves the way for further developments. We are interested in leveraging our earlier formalization of zero-knowledge protocols [5] to statistical zero-knowledge, and to use the result as a back-end for a certifying ZK compiler, in the style of [1]. We also intend to pursue the machine-checked formalization of indifferentiability proofs, and in particular to show that the finalists of NIST SHA-3 competition are indifferentiable from a random oracle. Finally, it would be of interest to enhance EasyCrypt [3], an automated front-end that generates verifiable security proofs in CertiCrypt, so that it can manipulate the notions of equivalence considered in this paper (and in [6]).

Acknowledgments. This work was partially funded by European Projects FP7-256980 NESSoS and FP7-229599 AMAROUT, Spanish project TIN2009-14599 DESAFIOS 10, Madrid Regional project S2009TIC-1465 PROMETIDOS and French project ANR SESUR-012 SCALP.

References

1. Almeida, J.B., Bangerter, E., Barbosa, M., Krenn, S., Sadeghi, A.-R., Schneider, T.: A Certifying Compiler for Zero-Knowledge Proofs of Knowledge Based on Σ-Protocols. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 151–167. Springer, Heidelberg (2010)
2. Audebaud, P., Paulin-Mohring, C.: Proofs of randomized algorithms in Coq. Sci. Comput. Program. 74(8), 568–589 (2009)
3. Barthe, G., Grégoire, B., Heraud, S., Zanella Béguelin, S.: Computer-Aided Security Proofs for the Working Cryptographer. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 71–90. Springer, Heidelberg (2011)
4. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Formal certification of code-based cryptographic proofs. In: 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, pp. 90–101. ACM, New York (2009)
5. Barthe, G., Hedin, D., Zanella Béguelin, S., Grégoire, B., Heraud, S.: A machine-checked formalization of Sigma-protocols. In: 23rd IEEE Computer Security Foundations Symposium, CSF 2010, pp. 246–260. IEEE Computer Society, Los Alamitos (2010)
6. Barthe, G., Köpf, B., Olmedo, F., Zanella Béguelin, S.: Probabilistic reasoning for differential privacy. In: 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012. ACM (2012)
7. Bellovin, S., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: 13th IEEE Symposium on Security and Privacy, S&P 1992, pp. 72–84. IEEE Computer Society, Los Alamitos (1992)
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17, 297–319 (2004)
9. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient Indifferentiable Hashing into Ordinary Elliptic Curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010)
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
11. Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. Journal of Computer Security 15(3), 321–371 (2007)

12. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Computer Security 18(6), 1157–1210 (2010)
13. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
14. Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
15. Farashahi, R.R., Fouque, P.A., Shparlinski, I., Tibouchi, M., Voloch, J.F.: Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. Mathematics of Computation (2011)
16. Fleischmann, E., Gorski, M., Lucks, S.: Some Observations on Indifferentiability. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 117–134. Springer, Heidelberg (2010)
17. Fouque, P.-A., Tibouchi, M.: Deterministic Encoding and Hashing to Odd Hyperelliptic Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 265–277. Springer, Heidelberg (2010)
18. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
19. Gonthier, G., Mahboubi, A., Rideau, L., Tassi, E., Théry, L.: A Modular Formalisation of Finite Group Theory. In: Schneider, K., Brandt, J. (eds.) TPHOLs 2007. LNCS, vol. 4732, pp. 86–101. Springer, Heidelberg (2007)
20. Hurd, J., Gordon, M., Fox, A.: Formalized elliptic curve cryptography. In: High Confidence Software and Systems, HCSS 2006 (2006)
21. Icart, T.: How to Hash into Elliptic Curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Heidelberg (2009)
22. Icart, T.: Algorithms Mapping into Elliptic Curves and Applications. Ph.D. thesis, Université du Luxembourg (2010)
23. Maurer, U.M., Renner, R.S., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
24. Pierro, A.D., Hankin, C., Wiklicky, H.: Approximate non-interference. Journal of Computer Security 12(1), 37–82 (2004)
25. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with Composition: Limitations of the Indifferentiability Framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011)
26. Segala, R., Turrini, A.: Approximated computationally bounded simulation relations for probabilistic automata. In: 20th IEEE Computer Security Foundations Symposium, CSF 2007, pp. 140–156 (2007)
27. Shallue, A., van de Woestijne, C.E.: Construction of Rational Points on Elliptic Curves over Finite Fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 510–524. Springer, Heidelberg (2006)
28. Shoup, V.: A Computational Introduction to Number Theory and Algebra, 2nd edn. Cambridge University Press (2009)
29. Smith, G.: On the Foundations of Quantitative Information Flow. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 288–302. Springer, Heidelberg (2009)
30. Théry, L., Hanrot, G.: Primality Proving with Elliptic Curves. In: Schneider, K., Brandt, J. (eds.) TPHOLs 2007. LNCS, vol. 4732, pp. 319–333. Springer, Heidelberg (2007)