Symbolic synthesis of indifferentiability attacks

Itsaka Rakotonirina
itsaka.rakotonirina@mpi-sp.org
MPI-SP

Miguel Ambrona
mac.ambrona@gmail.com

Alejandro Aguirre
alejandro@cs.au.dk
Aarhus University

Gilles Barthe
gjbarthe@gmail.com
MPI-SP, IMDEA Software Institute

ABSTRACT

We propose fully automated methods for synthesising attacks against indifferentiability, a powerful simulation-based notion of security commonly used to reason about symmetric constructions. Our methods are inspired from symbolic cryptography which is popular to reason about, e.g., cryptographic protocols. For that, we introduce a core programming language for *algebraic distinguishers* and study the class of *universal distinguishers*, who win the indifferentiability game against every simulator; then, we show that the universality of algebraic distinguishers can be reduced to solving systems of algebraic, deducibility and static-equivalence constraints.

Our approach is implemented in a tool, AutoDiff, which solves these constraint systems, and applies heuristics to automate the cryptanalysis (i.e., to search automatically for universal distinguishers). We evaluate the tool with many non-trivial attacks from the literature on Feistel networks and Even-Mansour blockciphers among others. Our tool is able to check the validity these attacks, and in many cases to synthesise them without guidance. To our knowledge, AutoDiff is the first practical tool for indifferentiability attacks.

CCS CONCEPTS

• **Security and privacy → Logic and verification**; *Block and stream ciphers*; *Cryptanalysis and other attacks*.

KEYWORDS

formal methods, computer-aided cryptography, indifferentiability

ACM Reference Format:
Itsaka Rakotonirina, Miguel Ambrona, Alejandro Aguirre, and Gilles Barthe. 2022. Symbolic synthesis of indifferentiability attacks. In *Proceedings of the 2022 ACM ASIA Conference on Computer and Communications Security (ASIACCS'22), May 30–June 3, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 15 pages.

1 INTRODUCTION

Algebraic cryptanalysis is a general set of techniques for reducing the (in)security of symmetric cryptographic constructions to solving systems of equations. However, the size and complexity of these equations make their manual resolution at best unwieldy. To address this bottleneck, cryptanalysts have built tools that search for attacks [Cou03, CM03, BPW06, BDF11, Leu12, DF16, SSD+18] and this approach has been successfully applied to find vulnerabilities against many block ciphers. However, these tools are focused on indistinguishability-based notions of security, and cannot be used for analysing other standard notions of security.

One such popular notion is *indifferentiability* [MRH04, RSS11], which was typically regarded as an important criterion for the NIST selection of SHA-3 [BDPVA08]. The prime purpose of indifferentiability is to support compositional analysis of cryptographic constructions. To this end, it is stated as a simulation-based notion of security, relating a real cryptographic component C to another ideal component R. Informally, it guarantees that C can be replaced by R in a larger system for the security analysis of a so-called *single-stage* security game. In practice, indifferentiability is defined through a game between a *distinguisher* D and a *real or ideal* oracle:

(1) the real oracle system is the actual component C built from a set of smaller ideal components Q, and
(2) the ideal system is a random function R and a simulator S that simulates the small components of Q by interacting with R.

The security game consists of the distinguisher attempting to correctly guess against which of the two systems it plays (see Figure 1). The classical notion of indifferentiability states that there is a simulator S, s.t. any distinguisher D can only get a negligible advantage in this game through a polynomial number of oracle calls.

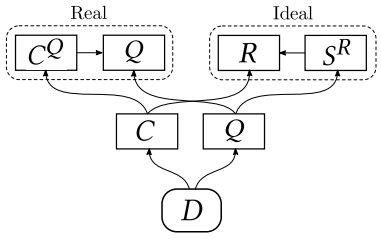


Figure 1: Indifferentiability security experiment

Indifferentiability is a common target property for symmetric cryptographic constructions, and has been widely studied for Feistel networks [Fei73], Even-Mansour block ciphers [EM97], confusion diffusion mechanisms [DSSL16], and Lai–Massey constructions. Unfortunately, (dis)proving indifferentiability of these constructions is tedious and error-prone, even more so than for the simpler indistinguishability-based definitions. On the provable security side for example, Coron *et al.* [CPS08] proved that the 6-round

ASIACCS'22, May 30–June 3, 2022, Nagasaki, Japan

Feistel network was indifferentiable from a random permutation; but Holensetein *et al.* [HKT10] pointed out two years later that the proof was invalid. Follow-up works attempted to remedy this flaw, resulting in a proof of indifferentiability for 8-round Feistel networks [DS16]. Similar challenges have been faced for other constructions, including Iterated Even-Mansour [LS13, DSST17].

On the cryptanalysis side, which is the focus of this paper, one usually considers the stronger notion of *universal attack*, which exhibit a distinguisher D that can distinguish between the real and ideal worlds for any possible simulator S. Proving that a construction admits a universal distinguisher is theoretically stronger than showing that it is not indifferentiable, yet most pen-and-paper attacks indeed exhibit a universal distinguisher. However, should it be for standard or universal differentiability, techniques from algebraic cryptanalysis do not appear as a natural fit: the simulation-based nature of the notion makes it unlikely to reduce the attack search to solving a simple system of algebraic equations. One key hurdle is to identify a richer set of constraints that can accommodate the universal quantification over all simulators.

In this paper, we put forward the use of richer constraints borrowed from *symbolic cryptography* [KM07, BCK09, BGJ+19]. This approach, which can be traced back to the seminal work of Dolev and Yao [DY81], uses a simplified model where cryptographic primitives are modelled algebraically, by means of equations that idealise their effects: for instance, encryption is typically modelled by two constructors for encryption and decryption, and an equation for encryption-decryption cancellation. The algebraic nature of the symbolic model fosters automation, as security analyses in this context are thus able to leverage standard techniques from unification theory such as *deducibility* or *static equivalence* as in [Bau07, CCD13, BGJ+19], whose decidability has been extensively studied [AC06, ACD07, BCLD07, CDK12].

Contributions

This paper presents AutoDiff, the first automated tool for checking and finding attacks (i.e., universal distinguishers) against the indifferentiability of cryptographic constructions. AutoDiff supports the validity of many non-trivial attacks from the literature, and also (re)discovers automatically several of them. For that:

(1) We introduce a syntax for algebraic distinguishers, a restricted class of distinguishers that encompasses many indifferentiability attacks from the literature;
(2) We show that for a candidate distinguisher expressed in our our syntax, universality can be reduced algorithmically to insolvability of a constraint system. They combine in a novel way standard notions of symbolic constraints, including algebraic equations but also *deducibility constraints* and *static equivalence*.
(3) Relying on a backend solving of our constraint systems, we develop heuristics searching for violations of indifferentiability automatically. We implemented a prototype, AutoDiff, and evaluated it on frameworks such as *Feistel networks*, *Even-Mansour ciphers*, or *Merkle-Damgård constructions*. Our results show that AutoDiff can verify many non-trivial attacks from the literature, and in many cases retrieve them.

Related work

Dolev-Yao models. Our work leverages many techniques initially designed for analysing cryptographic protocols in symbolic models. These techniques originate from the seminal work of Dolev and Yao [DY81] and are primarily characterised by an idealisation of the attacker's deducing capabilities, so that they can be described in purely algebraic terms. This algebraic view allows a direct connection to logic, which is highly beneficial for automation. The symbolic model has been used extensively for analysing cryptographic protocols, both for proving security and for discovering logical flaws; for some recent examples, see, e.g., [BBK17, BDH+18, CHH+17, JK21]. One common approach is to translate the protocol into a set of algebraic *intruder constraints* [Bau07, DLLT08, CR10, DKP12, CCD13, CKR18], hence reducing the insecurity problem to the verification of symbolic notions such as deducibility or static equivalence [AC06, BCLD07, CDK12]. We use the same notion of constraint, and leverage existing work on constraint solving [AC06, Bau07, ACD07, DLLT08, CR10, CD12, CDK12, DKP12].

Computational soundness. We also uses techniques of computational soundness, i.e., that relate security in the symbolic and computational models. This line of work, initiated by Abadi and Rogaway [AR07] and further developed in a long series of works, including [KM07, CLC08, BCK09, CKW11], heavily relies on some of the tools used by our approach. We use in particular the results of [BCK09]. Interestingly for our work, the usual theories of XOR (present in most of the constructions we analyse) are known to be incompatible with computational soundness in general, at least for an unbounded number of symbolic operations [Unr10]. This justifies in particular our restriction to distinguishers performing a bounded number of operations (which is a key assumption in the proof of our main theorem, and we show in Appendix 6.2 that a minimal unbounded extension of our framework is unsound).

Variations of the approach. Alternatives to computational soundness are symbolic methods for reasoning about computational security. They are often specialised to a specific problem: padding-based encryption [CDE+08, BCG+13], symmetric encryption [MKG14, HKM15, Mea20], pairing-based, orlattice-based security [BGS15, BFG+18]. These works have very different goals from ours (indistinguishability proofs vs. indifferentiability attacks), but there is partial overlap in the constraint solving techniques used in the backend—the closest relationship being with Meadows [Mea20]. Yet another approach similar to ours is what has been developed in the generic group model (GGM) and its variants [Nec94, Sho97, BBG05, Mau05]. They are, in short, idealised models to reason about lower and upper bounds of group-based algorithms, such as those to solve discrete logarithm. Typically, some works based on these models use constraints to automate analyses in the GGM [BFF+14, ABS16, ABGW17]: as us, they generate constraints that are then solved by general-purpose tools. However, the notion of constraint they rely on is simpler than ours, e.g., it does not involve static equivalence.

2 INDIFFERENTIABILITY FROM RANDOM

2.1 Algorithms

Formalism. Throughout the paper, we refer to the following notion of *(randomised) algorithm* $A : X \to Y$ to model program

executions. An algorithm A is specified as a probabilistic transition system $\rightarrow_A$ over a chosen set of states $\mathbf{S}$. This includes *input states* $\mathsf{input}(x)$, $x \in X$, and output states $\mathsf{output}(y)$, $y \in Y$. We only assume that for all states $s \in \mathbf{S}$, the set $\{s' \in \mathbf{S} \mid s \rightarrow_A s'\}$ of *successors* of s is finite (and empty *iff* s is an output state). An *execution* of $A(x)$ is then simply a sequence of transitions from $\mathsf{input}(x)$ to some $\mathsf{output}(y)$. In the vein of probabilistic Turing machines, algorithms are *randomised* in that the state following s in an execution is picked uniformly at random from all possible successors (independently of previous execution steps). The algorithm is *deterministic* if each state has at most one successor; note in particular that any function F can be interpreted as a trivial deterministic algorithm with transitions $\mathsf{input}(x) \rightarrow_F \mathsf{output}(F(x))$. The probability of an execution from $\mathsf{input}(x)$ reaching $\mathsf{output}(y)$ is then written

$$\Pr[A(x) = y] .$$

To model security games, we also consider algorithms that are parametrised by the execution of another one. For that we say that A has *oracle access* to a terminating algorithm R (which is made explicit with the notation A^R) if on certain states $\mathbf{Q} \subseteq \mathbf{S}$ called *query states*, A stops its normal execution, runs a probabilistic execution of some $R(x)$ instantly, and chooses its next state depending on the output state of R—after which the execution of A^R resumes.

Constructions. We specifically operate over the group $G = \{0,1\}^\eta$ of bitstrings of length η equipped with exclusive or (xor, $\oplus$). The integer η is called the *security parameter*, which is a caliber for probabilistic security properties. A function $F : G^p \rightarrow G^q$ is also called a *keyed permutation* when $p \geqslant q$ and there exists a function $F^{-1} : G^p \rightarrow G^q$, called its *inverse*, such that for all $\mathbf{x} \in G^q, \mathbf{y} \in G^{p-q}$,

$$F^{-1}(F(\mathbf{x},\mathbf{y}),\mathbf{y}) = \mathbf{x} \qquad \text{and} \qquad F(F^{-1}(\mathbf{x},\mathbf{y}),\mathbf{y}) = \mathbf{x} .$$

If Q is a set of functions, $Q^* \subseteq Q$ is the set of keyed permutations of Q. A *construction* C^Q is an algorithm with oracle access to the functions of Q, called *primitives*, and to the inverses F^{-1}, $F \in Q^*$. A primitive is always *ideal* (or *random*), i.e., it is formally a *distribution* of functions, here the uniform distribution over all functions, or all keyed permutations, of adequate domain. The effective oracle F is then sampled from this distribution. By abuse of vocabulary, we will talk about ideal functions as if they were actual functions.

2.2 The indifferentiability game

Our formalisation is based on [DS16]. *Indifferentiability* expresses the impossibility to tell apart a construction C^Q and an ideal function R with significant probability, using the following key notion:

Definition 2.1 (distinguisher). A *distinguisher* D is an algorithm that initiates a bounded number $q \in \mathbb{N}$ of queries to a collection of oracles (and possibly to their inverses for the keyed permutations among them), and eventually outputs 0 or 1.

In practice, the security experiment for indifferentiability is a game where a distinguisher is given oracle access to either:

(1) the construction C^Q and the ideal primitives Q (*real world*),
(2) or a random function R and a simulation of Q (*simulated world*),

and needs to guess which of the two oracle systems it interacts with. When it succeeds to do so, D is said to *win the indifferentiability game*. Rephrasing, for indifferentiability to hold, it should be possible to simulate the behaviour of the ideal primitives Q from a random function R, in a way that no distinguishers call tell a difference between the real construction and the simulated one.

Definition 2.2 (indifferentiability). Let C^Q be a construction and R be an ideal function (keyed permutation if C^Q is one). We say that C^Q is *indifferentiable* if there exist a polynomial $t_S(\eta, q)$, and $\varepsilon(\eta, q)$ negligible in η such that, for every $q \in \mathbb{N}$, there exists a randomised algorithm S^R (called a *simulator*) running in time $t_S(\eta, q)$ such that for all distinguishers D making at most q oracle queries,

$$\left|\Pr[D^{C^Q,Q} = 1] - \Pr[D^{R,S^R} = 1]\right| \leqslant \varepsilon(\eta, q)$$

where probabilities account for the sampling of ideal functions.

We recall that $f(\eta)$ being *negligible* means that for any $n \in \mathbb{N}$, there exists η_0 such that, for every $\eta \geqslant \eta_0$, $f(\eta) \leqslant \eta^{-n}$. On the contrary, $f(\eta)$ is *overwhelming* if $1 - f(\eta)$ is negligible. Note also that, in the above definition, the simulator is only tasked to simulate the oracle calls of D to the primitives Q, whereas the calls to the main oracle C^Q and its potential inverse $(C^{-1})^Q$ are automatically simulated by R and R^{-1}.

In the original definition of indifferentiability, simulators are stateful programs and may thus hold information from one query to another. To account for this fact in a lightweight manner, we assume for simplicity that, on every query, S implicitly receives the list of all past queries made to it.

2.3 Universal differentiability

Strictly speaking, disproving indifferentiability requires to show that, for all simulators S, there exists a distinguisher D telling apart the real and simulated worlds with significant probability. It is however more convenient to exhibit one *single* distinguisher D (qualified as *universal*) that has a non-negligible advantage in the security game against *all* possible simulators S.

Definition 2.3 (universal distinguisher). A distinguisher D is said to be *universal* against a construction C^Q if, for all simulators S running in polynomial time in the security parameter η, we have for some non-negligible function $\varepsilon : \mathbb{N} \rightarrow \mathbb{R}$:

$$\left|\Pr[D^{C^Q,Q} = 1] - \Pr[D^{R,S^R} = 1]\right| \geqslant \varepsilon(\eta)$$

Proposition 2.4 (universal differentiability). *If there exists a universal distinguisher D against a construction C^Q, then C^Q is differentiable from a random function.*

In theory, this characterisation is a sufficient but not necessary condition. There might exist constructions that are not indifferentiable from a random function, but that cannot be differentiated by any fixed algorithm D. To the best of our knowledge however, the existence of such constructions is an open problem. In practice, the standard approach for disproving indifferentiability therefore remains to exhibit a universal distinguisher, as evidenced by attacks on various high-profile block ciphers such as 5-round Feistel networks [CHK+16], 4-round iterated Even-Mansour ciphers [DSST17], Merkel-Damgård or 2-round confusion-diffusion networks [DSSL16]. Our goal is to automate the verification of universality for a class of distinguishers encompassing these attacks.

2.4 Examples of constructions

The two simplest constructions we can conceive are the following ones, given a primitive $F : G \to G$:

$$C_1(x) = F(x) \qquad C_2(x) = x$$

The construction C_1 is by definition an ideal function: no distinguisher can win the security game against the simulator that simulates a query to $F(x)$ by a call to $R(x)$. On the contrary, C_2 is trivially differentiable: consider the distinguisher D that (1) samples a group element $x \in G$ uniformly at random; (2) calls the main oracle on x, and thus obtains $y = x$ in the real world and $y = R(x)$ in the simulated world; and (3) returns 1 *iff* $x = y$. Then D wins the security game with overwhelming probability against any simulator S. Let us now introduce more complex constructions that will serve as running examples throughout the paper and primary targets of interest for our prototype analyser.

Feistel networks. A k-round Feistel network [Fei73] is a symmetric block cipher $C : G^2 \to G^2$ illustrated in Figure 2 and used among others in standards such as PKCS [BR94, FOPS01].

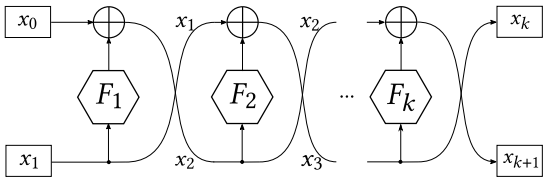


Figure 2: k-round Feistel network

The construction receives the inputs (x_0, x_1) and returns the outputs (x_k, x_{k+1}) by iterating the following computation:

$$x_{i+1} = x_{i-1} \oplus F_i(x_i),$$

using the primitives $F_i : G \to G$, $i \in [1, k]$. For instance, for 3 rounds, the first output of C is $x_3 = x_1 \oplus F_2(x_0 \oplus F_1(x_1))$.

Feistel networks have the convenient property of being invertible due to the cancellation property of $\oplus$, and are therefore used to build invertible pseudorandom permutations from (non-invertible) pseudorandom primitives. One desired security property is hence that their structure preserves the randomness of the atomic functions F_i, which is formalised by the indifferentiability of C.

Even-Mansour ciphers. A k-round (iterated) Even-Mansour cipher [EM97], a symmetric block cipher $C : G^2 \to G$ (Figure 3). On inputs (x, y), the construction performs k applications of primitives $P_i : G \to G$ to x, each preceded and followed by a mask using a same key y. The primitives P_i are assumed to be invertible.

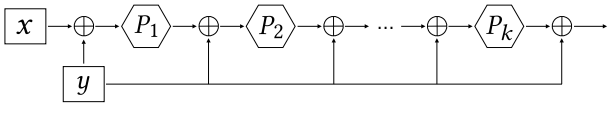


Figure 3: k-round Iterated Even-Mansour cipher

The overall construction C is therefore a keyed permutation. That is, we can define $C^{-1} : G^2 \to G$ such that for all $x, y \in G$,

$$C(C^{-1}(x, y), y) = x \qquad C^{-1}(C(x, y), y) = x$$

3 ALGEBRAIC DISTINGUISHERS

The distinguishers we consider will be restricted to operations that can be abstracted by uninterpreted symbols and algebraic equations. This way, negligible biases in probabilistic events can be filtered out from analyses, making automation simpler. The resulting formalism borrows much from the applied pi-calculus [ABF17], a symbolic model often used in the analysis of cryptographic protocols.

3.1 Symbolic model of group computations

We recall that we focus on constructions on the group $G = \{0,1\}^\eta$ of η-bitstrings. We model them symbolically by a set of *atomic data*:

$$\mathcal{A} = \mathsf{Cst} \uplus \mathsf{Rnd} \uplus \mathsf{Vars}^1$$

Elements of Rnd (*random group elements*) represent values sampled by the distinguisher (and therefore cannot be guessed by the simulator with non-negligible probability). On the contrary, elements of Cst (*constant group elements*) model fresh values that the simulator may generate during its simulation. Elements of Vars^1 (*first-order variables*, or simply *variables*) are used as placeholders for other computations. A term without variables is called *ground*. Operations over atomic data are then modelled by *function symbols* F, that are abstract uninterpreted symbols tagged with an *in-arity* $p > 0$ and an *out-arity* $q > 0$ indicating how many arguments they take and return. We write $F : G^p \to G^q$ to express this fact. These symbols are then gathered into so-called *signatures*:

$$\mathcal{F} = \mathsf{Fun} \uplus \mathsf{Orcl}\,.$$

Symbols of Fun model the operations such as $\oplus$ used for building the construction and distinguishers, whereas symbols of Orcl, called *oracle symbols*, model the various oracles of the security game.

Definition 3.1 (first-order term). Lists of first-order terms $\mathbf{t}$ (or simply "*terms*" for short) are obtained by applying function symbols to atomic values or other terms, accordingly to their arities. We also extend the notion of arity through the following inference rules:

$$\frac{x \in \mathcal{A}}{x : G} \qquad \frac{\mathbf{t}_1 : G^p \qquad \mathbf{t}_2 : G^q}{\mathbf{t}_1, \mathbf{t}_2 : G^{p+q}} \qquad \frac{\mathbf{t} : G^p \qquad F : G^p \to G^q}{F(\mathbf{t}) : G^q}$$

We write $\mathcal{T}(S)$ the set of terms $\mathbf{t} : G^p$, $p \in \mathbb{N}$, constructable from the symbols and atomic data of $S \subseteq \mathcal{F} \cup \mathcal{A}$. Similarly to function symbols, p is called the *arity* of $\mathbf{t} : G^p$.

First-order terms are classically used in symbolic models to represent computations [ABF17]. Here they specifically model those of the distinguisher: for example if the symbol $\mathbf{C} : G \to G$ stands for the main oracle of the security game, $\oplus : G^2 \to G$ for xor (with infix notation), and $x, y \in \mathsf{Vars}^1$, the term $\mathbf{C}(x) \oplus y : G$ models a call to the main oracle on x, summed with y. The functional properties of these symbols (e.g., commutativity of xor) are then modelled by a classical notion of *equational theory* below. First, we call a *substitution* σ a mapping from variables to terms of arity 1, written

$$\sigma = \{x_1 \mapsto \sigma(x_1), \ldots, x_p \mapsto \sigma(x_p)\}$$

and homomorphically extended to functions from terms to terms. We may write it in the more succinct manner $\sigma = \{\mathbf{x} \mapsto \mathbf{t}\}$ with $\mathbf{x} = x_1, \ldots, x_p$ and $\mathbf{t} = \sigma(x_1), \ldots, \sigma(x_p)$, and note $dom(\sigma) = \{\mathbf{x}\}$ the domain of σ. We also often use the standard postfix notation $t\sigma$ instead of $\sigma(t)$. This permits us to define:

Definition 3.2 (theory). An *equational theory* $\mathcal{E}$ over a signature $\mathcal{F}$ is a set of pairs $u, v \in \mathcal{T}(\mathsf{Vars}^1 \cup \mathsf{Fun})$, u and v of same arity, called *equations* and written $u =_{\mathcal{E}} v$. The relation $=_{\mathcal{E}}$ is closed by substitution and context, that is, it is extended to the smallest equivalence relation containing $\mathcal{E}$ and verifying the inference rules:

$$\frac{u =_{\mathcal{E}} v}{u\sigma =_{\mathcal{E}} v\sigma} \qquad \frac{\mathbf{t}_1 =_{\mathcal{E}} \mathbf{u}_1 \qquad \mathbf{t}_2 =_{\mathcal{E}} \mathbf{u}_2}{\mathbf{t}_1, \mathbf{t}_2 =_{\mathcal{E}} \mathbf{u}_1, \mathbf{u}_2} \qquad \frac{\mathbf{t} =_{\mathcal{E}} \mathbf{u}}{F(\mathbf{t}) =_{\mathcal{E}} F(\mathbf{u})}$$

In this paper, the pair $(\mathcal{F}, \mathcal{E})$ is called a *theory*.

3.2 Reference theory

From now on, we operate within what we call the *reference theory*, defined in Figure 4. It models the security game for a construction $C^Q : G^{p_{in}} \to G^{p_{out}}$, that is assumed to be definable in the reference theory, that is, as a function from terms to terms. Intuitively, the function symbols of the theory (Fun) model xor and projections, while its oracle symbols (Orcl) represent primitives of Q (interpreted as function symbols), the main oracle $\mathbf{C}$ (i.e., C^Q in the real world and an ideal function R in the simulated world), and their potential inverses referred to as specific symbols F^{-1}. In particular, we define

$$\mathsf{Orcl}_0 = Q \cup \{\mathbf{C} : G^{p_{in}} \to G^{p_{out}}\}$$

the set of the direct oracles of the security game, and Orcl is obtained in Figure 4 by adding inverse symbols to it. By abuse of notations, given $E \subseteq \mathsf{Orcl}_0$, we write E^* the set of the symbols of E corresponding to keyed permutations.

$$\mathsf{Fun} = \{0 : G, \oplus : G^2 \to G\} \cup \{\pi_{i/p} : G^p \to G \mid 0 < i \leqslant p\}$$
$$\mathsf{Orcl} = \mathsf{Orcl}_0 \cup \{F^{-1} \mid F \in \mathsf{Orcl}_0^*\}$$
$$x \oplus 0 =_{\mathcal{E}} x \qquad x \oplus x =_{\mathcal{E}} 0 \qquad x \oplus y =_{\mathcal{E}} y \oplus x$$
$$(x \oplus y) \oplus z =_{\mathcal{E}} x \oplus (y \oplus z)$$
$$\pi_{i/p}(x_1, \ldots, x_p) =_{\mathcal{E}} x_i \quad \text{for } i, p \in \mathbb{N}^*, i \leqslant p$$
$$\pi_{1/q}(F(\mathbf{x})), \ldots, \pi_{q/q}(F(\mathbf{x})) =_{\mathcal{E}} F(\mathbf{x}) \quad \text{for } (F : G^p \to G^q) \in \mathsf{Orcl}$$
$$F^{-1}(F(\mathbf{x}, \mathbf{y}), \mathbf{y}) =_{\mathcal{E}} \mathbf{x} \quad \text{for } F \in \mathsf{Orcl}_0^*$$
$$F(F^{-1}(\mathbf{x}, \mathbf{y}), \mathbf{y}) =_{\mathcal{E}} \mathbf{x} \quad \text{for } F \in \mathsf{Orcl}_0^*$$

Figure 4: Reference theory of the paper

3.3 Specifying algebraic distinguishers

Syntax. We now define a grammar for distinguishers that only perform computations that can be expressed in the reference theory:

D ::=	$D + D$		convex combination
	P		program
P ::=	$\mathbf{x} \leftarrow F(\mathbf{t});\ P$		query
	return φ		return
φ ::=	$u =^? v$	$\varphi \wedge \varphi$ $\neg\varphi$	test

where $(F : G^p \to G^q) \in \mathsf{Orcl}$, $\mathbf{x}$ is a list of q distinct variables, $\mathbf{t}, u, v \in \mathcal{T}(\mathsf{Fun} \cup \mathsf{Vars}^1 \cup \mathsf{Rnd})$ with $\mathbf{t} : G^p$ and u, v are of same arity. For succinctness we write $u \neq^? v$ instead of the formula $\neg(u =^? v)$.

Definition 3.3 (distinguisher code). A *distinguisher code* c is an object derived from the token D in the above grammar. We say that c is *non-branching* if it does not contain + operators.

Intuitively, $\mathbf{x} \leftarrow F(\mathbf{t})$ is used to query the oracle F on some inputs $\mathbf{t}$. The term $\mathbf{t}$ may contain random group elements (Rnd), modelling the uniform sampling of some bitstrings. The final return φ evaluates a test over bitstrings as the output of the algorithm (1 if satisfied, 0 otherwise). Finally, the probabilistic branching operator $c_1 + c_2$ executes either c_1 or c_2 with probability $\frac{1}{2}$. This introduces uncertainty on the simulator's side, which receives queries without knowing in general which branch is effectively being executed.

The syntax is presented in a theoretically-minimal form to lighten the framework, but convenient syntax extensions can easily be encoded. For instance, branching may be allowed not only at toplevel but at any point of the code, encoding $q; (D + D)$ by $(q; D) + (q; D')$. Another common mechanism is the use of instructions assert $\varphi; c$ that start building the final test of the distinguisher during the code.

Semantics. We now formalise how to interpret a code c as an actual distinguisher, that is, an algorithm $D = [\![c]\!]$ returning values of $\{0, 1\}$. We make explicit which oracles D may take by writing explicitly $[\![c]\!]^\omega$, where ω is called an *oracle system*, that is a token

$$\omega \in \{\mathsf{real}, \mathsf{sim}(S) \mid S \text{ simulator}\}$$

indicating in which world the code c is being executed. Referring to Section 2.1, we thus have to define a transition system $\to_D$ modelling the execution of the code c with oracle system ω. Its set of states $\mathbf{S}$ will contain the following elements:

$\mathsf{input}(c)$	c code
$\mathsf{output}(b)$	$b \in \{0, 1\}$
(c, σ)	c code, $\sigma : \mathsf{Vars}^1 \cup \mathsf{Rnd} \to G$

In a state (c, σ), the code c simply indicates the remaining instructions to execute. In particular, the query states are those of the form $(\mathbf{x} \leftarrow F(\mathbf{t}); c, \sigma)$. The mapping σ models the execution *store*, that is, it indicates to which bitstring corresponds each symbolic atomic data. We will allow our distinguisher to have access to a family of algorithms implementing the oracle system ω, as defined below.

Definition 3.4 (implementation). A family of algorithms $O = \{O_g\}_{g \in \mathsf{Orcl}}$ *implements* the oracle system ω when each O_g operates on tuples of bitstrings accordingly to the arity of g, and:

(1) if $\omega = \mathsf{real}$ then it holds that:
   (a) $\forall F \in \mathsf{Orcl}_0^*$, O_F is a keyed permutation and $O_F^{-1} = O_{F^{-1}}$;
   (b) $O_{\mathbf{C}} = C^{\{O_F \mid F \in Q\}}$;
(2) if $\omega = \mathsf{sim}(S)$, writing $Q' = Q \cup \{F^{-1} \mid F \in Q^*\}$, we derive from S a collection of simulators S_F, respectively answering to queries to $F \in Q'$, and it should hold that:
   (a) $\forall F \in Q'$, $O_F = S_F^{O_{\mathbf{C}}}$;
   (b) if C^Q is a keyed permutation, then $O_{\mathbf{C}}$ is a keyed permutation as well and $O_{\mathbf{C}}^{-1} = O_{\mathbf{C}^{-1}}$.

Putting everything together, we define in Figure 5 the transition relation on states that model the executions of distinguisher codes. The figure uses the following notations:

- given n variables $\mathbf{x} = x_1, \ldots, x_n$ and n bitstrings $\mathbf{t} = t_1, \ldots, t_n$, $\sigma[\mathbf{x} \mapsto \mathbf{t}]$ is the store that coincide with σ on $dom(\sigma) \setminus \{\mathbf{x}\}$ and that maps each x_i to t_i;
- if $t \in \mathcal{T}(\mathsf{Vars}^1 \cup \mathsf{Rnd} \cup \mathsf{Fun})$, $t\sigma$ refers to the bitstring obtained by applying σ homomorphically to t, interpreting the functions of Fun in the natural way as functions over bitstrings;
- if φ is formula (as those of the return statements of distinguishers), $\varphi\sigma \in \{0,1\}$ is the result of the evaluation of φ (1 meaning "satisfied"), interpreting $=^?$ as bitstring equality.

$$\mathsf{input}(c) \to (c, \sigma) \qquad \text{(Init)}$$

$$(c_1 + c_2,\ \sigma) \to (c_1, \sigma) \qquad \text{(Left)}$$

$$(c_1 + c_2,\ \sigma) \to (c_2, \sigma) \qquad \text{(Right)}$$

$$(\mathbf{x} \leftarrow F(\mathbf{t}); c,\ \sigma) \to (c, \sigma[\mathbf{x} \mapsto O_F(\mathbf{t}\sigma)]) \qquad \text{(Query)}$$

$$(\mathsf{return}\ \varphi, \sigma) \to \mathsf{output}(\varphi\sigma) \qquad \text{(Return)}$$

Figure 5: Semantics of the code c with oracle access to O

Intuitively, Rule (Init) samples a store, thus modelling the fact that random group elements are sampled uniformly at random. Rules (Left) and (Right) simply execute one branch among the possible two with probability $\frac{1}{2}$. Rule (Query) executes a query by calling to the oracles initialised consistently by Rule (Init), and finally Rule (Return) concludes the algorithm by evaluating the test φ in the store as expected. All in all we obtain:

Definition 3.5 (code semantics). Given a distinguisher code c, we write $[\![c]\!]$ for the corresponding distinguisher, i.e., the algorithm that, with access to a family of oracles O, is induced by the transition system of Figure 5. Given $b \in \{0,1\}$ and ω oracle system, we write:

$$\Pr[[\![c]\!]^{\omega} = b]$$

to refer to the probability $\Pr[[\![c]\!] = b]$ where $[\![c]\!]$ has oracle access to a family O implementing ω sampled uniformly at random from the (finite) set of all such families. This sampling of O intuitively corresponds to the sampling of ideal functions in the security game.

Algebraic distinguishers. Now that we have a formal syntax and semantics for specifying distinguishers, we have characterised the class of distinguishers we plan to study in this paper.

Definition 3.6 (algebraic). A distinguisher D is *algebraic* if $D = [\![c]\!]$ for some code c and $\Pr[[\![c]\!]^{\mathsf{real}} = 1]$ is overwhelming.

Proposition 3.7 (Algebraic differentiability). *If $D = [\![c]\!]$ is an algebraic distinguisher, the following points are equivalent:*

(i) D is universal
(ii) for all simulators S running in polynomial time in the security parameter η, $\Pr[[\![c]\!]^{\mathsf{sim}(S)} = 1]$ is non-overwhelming

Example 3.8. Let us go back to the running example of Feistel networks we introduced in Section 2.4. Consider the case where C is a 3-round network. Figure 6 presents two distinguisher's codes against C, Daisy and David. The only difference between the two is the order of their queries.

Daisy:

1 $x_1 \leftarrow F_1(r_1)$
2 $x_2 \leftarrow F_2(r_2)$
3 $z, z' \leftarrow \mathbf{C}(x_1 \oplus r_2, r_1)$
4 return $z =^? r_1 \oplus x_2$

David:

1 $x_2 \leftarrow F_2(r_2)$
2 $x_1 \leftarrow F_1(r_1)$
3 $z, z' \leftarrow \mathbf{C}(x_1 \oplus r_2, r_1)$
4 return $z =^? r_1 \oplus x_2$

with $r_1, r_2 \in \mathsf{Rnd}$

Figure 6: Two distinguishers for 3-round Feistel

First, we observe that the final test $\varphi = (z =^? r_1 \oplus x_2)$ holds in the real world with probability 1 since, by definition of Feistel networks, if we write $x_i = F_i(r_i)$ and $(z, z') = C^Q(x_1 \oplus r_2, r_1)$:

$$\begin{aligned} z &= r_1 \oplus F_2(x_1 \oplus r_2 \oplus F_1(r_1)) \\ &= r_1 \oplus F_2(r_2) \\ &= r_1 \oplus x_2\,. \end{aligned}$$

In particular, both Daisy and David are algebraic. However, only the latter is universal: using the characterisation of Proposition 3.7, it means that there exists a simulator, *Susan*, that can answer Daisy's queries in a way that φ holds. Rephrasing, Susan has to compute, with access to a random permutation $R : G^2 \to G^2$, two bitstrings x_1 and x_2 that verify the relation

$$\pi_{1/2}(R(x_1 \oplus r_2, r_1)) = r_1 \oplus x_2 \qquad (\star)$$

where r_1, r_2 are two values sampled uniformly at random by Daisy and David. In particular, Susan has a negligible probability of guessing r_1, r_2, *but* obtains their values through queries. The only difference between Daisy and David is then what queries have effectively been sent to Susan at the time she has to compute x_1 and x_2. For example, when interacting with Daisy, Susan has already received r_1 (argument of the first query) and r_2 (argument of the second query) at the time she has to compute x_2. On the contrary, she only received r_2 when playing against David. It is then sufficient to observe that, intuitively, it is not possible to compute a value x_2 that satisfies relation $(\star)$ without knowing the values of both r_1 and r_2: this explains why Susan can win against Daisy but not David. The next section formalises this intuition.

4 SYMBOLIC ANALYSIS OF UNIVERSALITY

We now characterise simulator's constraints, thus formalising the concepts outlined in Example 3.8. This is inspired from standard notions in symbolic frameworks [Bau07, DLLT08, CCD13, CKR18].

4.1 Constraint systems

First-order terms were used to model the distinguisher's computations; we introduce now a refined notion of *second-order* terms, expressing the specific restrictions of the simulator (such as not being able to guess random group elements sampled by the distinguisher). For that we first define *frames*, modelling the information available to the simulator as a finite set of entries.

Definition 4.1 (frame). A *frame* is a substitution of the form

$$\Phi = \{\mathsf{ax}_1 \mapsto \mathbf{t}_1, \ldots, \mathsf{ax}_p \mapsto \mathbf{t}_p\}$$

where the domain of Φ consists of special variables from a dedicated set $\mathsf{Axioms} = \{\mathsf{ax}_i\}_{i \in \mathbb{N}}$ of so-called *axioms*, and the $\mathbf{t}_i$'s are terms.

As intuited above, frames model the simulator's knowledge. Consider for example a distinguisher sampling a bitstring r and sending the query $x \leftarrow F(r)$ to the simulator, which therefore receives the value of r. This is modelled by an entry $\mathsf{ax}_i \mapsto (\bar{F}, r)$ recorded in a frame, for some $\bar{F} \in \mathsf{Cst}$ identifying the primitive used. Thus, although the simulator cannot guess the value of r *a priori*, it will be able to access it by reference, using the computation $\pi_{2/2}(\mathsf{ax}_i)$. More generally, the simulator's computations are modelled by:

Definition 4.2 (second-order terms). If we consider the set $\mathit{Priv} = Q \cup \{F^{-1} \mid F \in Q^*\} \cup \mathsf{Rnd}$, a *second-order term* ξ is a term of

$$\mathcal{T}((\mathcal{F} \cup \mathcal{A} \cup \mathsf{Axioms}) \setminus \mathit{Priv}) .$$

We also consider a set of so-called *second order variables* $\mathsf{Vars}^2 = \{X, Y, Z, \ldots\}$, where each $X \in \mathsf{Vars}^2$ has a *multiplicity* $i \in \mathbb{N}$, which is sometimes made explicit by writing $X : i$.

A second-order variable $X : i$ is thus a placeholder for a second-order term—that is, a simulator's computation—that may only use the first i values sent by the distinguisher. Consistently, we call a *second-order substitution* Σ a mapping from second-order variables to second-order terms in a way that respects multiplicities, that is, for all $(X : i) \in \mathit{dom}(\Sigma)$, $X\Sigma$ only contains axioms from $\{\mathsf{ax}_1, \ldots, \mathsf{ax}_i\}$. Using all of this, we can finally define *constraint systems*, that gather a frame (modelling the aggregated knowledge of the simulator), deducibility constraints (expressing the simulator's ability to compute the answers it gives to queries), and a formula φ (indicating the relations that the simulated terms should verify).

Definition 4.3 (constraint system). A *constraint system* is a tuple $\Gamma = (\Phi, \mathsf{D}, \varphi)$, where Φ is a frame, φ is a formula (as those used in the return instructions of distinguisher codes), and D is a set of so-called *deducibility constraints* $X \vdash^? \mathbf{x}$ ($X \in \mathsf{Vars}^2$, $\mathbf{x} \in \mathcal{T}(\mathsf{Vars}^1)$).

Definition 4.4 (solution). A *solution* Σ of a constraint system $(\Phi, \mathsf{D}, \varphi)$ is a second-order substitution such that there exists a substitution σ (called a *first-order solution* of Σ) such that:

(1) Σ *computes* σ: $\quad \forall (X \vdash^? \mathbf{x}) \in \mathsf{D},\ X\Sigma\Phi =_{\mathcal{E}} \mathbf{x}\sigma$
(2) *the formula is satisfied:* $\quad \varphi\sigma$ holds when interpreting $=^?$ as $=_{\mathcal{E}}$

Example 4.5. Consider again our running example, and in particular, the distinguisher Daisy and David introduced in Example 3.8. The task of a simulator playing against them is to answer to the two queries in a way that the final test φ holds:

$$\pi_{1/2}(\mathbf{C}(x_1 \oplus r_2, r_1)) =^? r_1 \oplus x_2 \qquad (\varphi)$$

This is intuitively expressed by the two constraint systems $\Gamma_{\mathsf{Daisy}} = (\Phi, \mathsf{D}, \varphi)$ and $\Gamma_{\mathsf{David}} = (\Phi', \mathsf{D}', \varphi)$, with $\bar{F} \in \mathsf{Cst}$ and:

$$\Phi = \{\mathsf{ax}_1 \mapsto (\bar{F}, r_1), \mathsf{ax}_2 \mapsto (\bar{F}, r_2)\} \qquad \mathsf{D} = \{X{:}1 \vdash^? x_1, Y{:}2 \vdash^? x_2\}$$
$$\Phi' = \{\mathsf{ax}_1 \mapsto (\bar{F}, r_2), \mathsf{ax}_2 \mapsto (\bar{F}, r_1)\} \qquad \mathsf{D}' = \{X{:}1 \vdash^? x_2, Y{:}2 \vdash^? x_1\}$$

The system Γ_{David} has no solutions: computing a suitable value of x_2 requires to have access to the values of both x_1 and x_2, whereas Γ_{David} requires to compute x_2 only from r_2. This reflects the fact that David is universal. On the contrary, Γ_{Daisy} has the following solution, thus describing how a simulator can win against Daisy, given an arbitrary constant $c \in \mathsf{Cst}$:

$$\Sigma = \{X \mapsto c, Y \mapsto \pi_{2/2}(\mathsf{ax}_1) \oplus \pi_{1/2}(\mathbf{C}(c \oplus \pi_{2/2}(\mathsf{ax}_2), \pi_{2/2}(\mathsf{ax}_1)))\}$$

4.2 Consistent solutions

So far we defined a notion of constraint system modelling the task of the simulator against a given distinguisher. But against a branching distinguisher $D = D_1 + \cdots + D_n$, the simulator has to win against *all* branches D_i; that is, D is universal *iff* the constraint systems that we will associate to each D_i have no solutions. However, we also have to account for the simulator's uncertainty about the executed branch. Indeed, consider for example, for $r, s \in \mathsf{Rnd}$:

$$D = x \leftarrow F(r);\ y \leftarrow F(s);\ P$$
$$D' = x \leftarrow F(s);\ y \leftarrow F(r \oplus s);\ P'$$

From the point of view of the simulator, the first two queries of D and D' both appear as F-evaluations on two random group elements. When playing against $D + D'$, the simulator hence cannot infer from them, with significant probability, whether it is interacting with the branch D or D'. It will in particular have to answer identically when interacting with either of them. Exploits of this mechanism can be found, e.g., in the attack on 4-round Even-Mansour ciphers from [DSST17]. We formalise this notion of branch indistinguishability by the classical notion of *static equivalence* [AC06].

Definition 4.6 (static equivalence). Let Φ and Ψ be two frames. We say that they are *statically equivalent*, written $\Phi \sim \Psi$, if

(1) $\mathit{dom}(\Phi) = \mathit{dom}(\Psi)$
(2) for all $\mathsf{ax}_i \in \mathit{dom}(\Phi)$, if $\mathsf{ax}_i\Phi : G^p$ then $\mathsf{ax}_i\Psi : G^p$
(3) for all second order terms ξ, ζ, we have $\xi\Phi =_{\mathcal{E}} \zeta\Phi$ *iff* $\xi\Psi =_{\mathcal{E}} \zeta\Psi$

Example 4.7. In the distinguishers D and D' described above, the simulator cannot distinguish between the first two queries of the two branches, which is modelled by the static equivalence of

$$\Phi_D = \{\mathsf{ax}_1 \mapsto (\bar{F}, r), \mathsf{ax}_2 \mapsto (\bar{F}, s)\}$$
$$\Phi_{D'} = \{\mathsf{ax}_1 \mapsto (\bar{F}, r), \mathsf{ax}_2 \mapsto (\bar{F}, r \oplus s)\}$$

where $\bar{F} \in \mathsf{Cst}$ is a token modelling that a query to the function F has been made. If we consider, instead, a distinguisher D_0 whose first two queries are $F(r)$ and $P(s)$ for some other primitive $P : G \to G$, it would corresponds to the frame:

$$\Phi_{D_0} = \{\mathsf{ax}_1 \mapsto (\bar{F}, r), \mathsf{ax}_2 \mapsto (\bar{P}, s)\}$$

This frame is not statically equivalent to Φ_D nor $\Phi_{D'}$. Indeed, if we take $\xi = \pi_{1/2}(\mathsf{ax}_2)$ and $\zeta = \bar{F}$ in the definition, the test "$\xi = \zeta$" holds in Φ_D and $\Phi_{D'}$ but not in Φ_{D_0}. This models the fact that, after receiving two queries from $D + D' + D_0$, a simulator can know whether it is playing against $D + D'$ or D_0. Consider then that the third queries of D and D' are, respectively, $F(t)$ and $F(s)$ for some fresh random group element t. This leads to the extended frames:

$$\Phi_D^+ = \Phi_D \cup \{\mathsf{ax}_3 \mapsto (\bar{F}, t)\}$$
$$\Phi_{D'}^+ = \Phi_{D'} \cup \{\mathsf{ax}_3 \mapsto (\bar{F}, s)\}$$

which are not statically equivalent any more. Indeed, the test

$$\pi_{2/2}(\mathsf{ax}_1) = \pi_{2/2}(\mathsf{ax}_2) \oplus \pi_{2/2}(\mathsf{ax}_3)$$

holds in $\Phi_{D'}^+$ but not in Φ_D^+. This formalises the fact that a simulator having received its third query from $D + D'$ can use this equality test to infer whether it is playing against D or D'.

To reflect this in the model, we finally introduce an original notion of *solution consistency*: it formalises this idea that indistinguishable branch prefixes should be treated identically by simulators.

Definition 4.8 (consistency). Let $\Gamma_1, \ldots, \Gamma_n$ be constraint systems, with $\Gamma_i = (\Phi_i, \mathsf{D}_i, \varphi_i)$ and Σ a common solution to all of them with respective first-order solutions σ_i. In the following we write $\Phi[\ell]$ the frame Φ restricted to its first ℓ axioms. We say that Σ is *consistent* if, for all $i, j \in [1, n]$, and for all $X : \ell \in \mathsf{D}_i, Y : \ell \in \mathsf{D}_j$,

$$\Phi_i[\ell]\sigma_i \sim \Phi_j[\ell]\sigma_j \quad \Rightarrow \quad X\Sigma = Y\Sigma \ .$$

4.3 Main result

We have gathered all ingredients to formalise our main theorem. As in Example 4.7, we associate a constant $\bar{F}$ to each oracle symbol $F \in \mathsf{Orcl}$, so that we can store in frames which oracle calls have been performed. Given a non-branching distinguisher code c,

$$\Gamma(c) = \mathsf{Transl}(\emptyset, \emptyset, c)$$

will refer to the translation of c into a constraint system, where:

$$\mathsf{Transl}(\Phi, \mathsf{D}, \mathsf{return}\ \varphi) = (\Phi, \mathsf{D}, \varphi)$$

$$\mathsf{Transl}(\Phi, \mathsf{D},\ x_1, \ldots, x_q \leftarrow F(\mathbf{t}); c) = \mathsf{Transl}(\Phi, \mathsf{D}, c\sigma)$$
$$\text{if } F \in \mathsf{Orcl} \setminus Q, \text{ and } \sigma = \{x_i \mapsto \pi_{i/q}(F(\mathbf{t}))\}_{i=1}^{q}$$

$$\mathsf{Transl}(\Phi, \mathsf{D},\ \mathbf{x} \leftarrow F(\mathbf{t}); c) = \mathsf{Transl}(\Phi', \mathsf{D} \cup \{X \vdash^{?} \mathbf{x}\}, c)$$
$$\text{if } F \in Q, \Phi' = \Phi \cup \{\mathsf{ax}_{|dom(\Phi)|+1} \mapsto \bar{F}, \mathbf{t}\}, \text{ and } X : |dom(\Phi')| \text{ fresh}$$

We assumed for simplicity that variables are adequately alpha-renamed so that variables are only bound once by queries. Under this assumption, our main result is stated as follows:

Main theorem. *Let $D = [\![c_1 + \cdots + c_n]\!]$ be an algebraic distinguisher for some non-branching distinguisher codes c_i. Then the following points are equivalent:*

(i) D is universal
(ii) the constraint systems $\Gamma(c_1), \ldots, \Gamma(c_n)$ have no common, consistent solutions Σ

We provide a formal proof of this result in the appendix, but give below an outline of the arguments we use.

Step 1: Reducing the class of simulators. The first step of our appendix proof is to reduce the problem to the study of simulators that are deterministic and perform a bounded number of oracle queries. This is a standard reduction relying on the fact that if a probabilistic simulator S wins with overwhelming probability against a distinguisher D, then in particular at least one execution of S wins against D with higher probability.

Step 2: Soundness of the reference theory. We show that if two ground terms are equal (resp. different) in the equational theory, then their interpretation as bitstrings are equal (resp. different) with overwhelming probability. Similarly, we show that if two frames are statically equivalent then they induce indistinguishable distributions. We obtain this convenient correspondence between the symbolic and computational frameworks by using characterisations of *computational soundness* found in [BCK09]. This justifies among other things the implication $(i) \Rightarrow (ii)$ of the main theorem. Specifically, the computational indistinguishability of statically equivalent frames justifies the consistency requirement of the theorem.

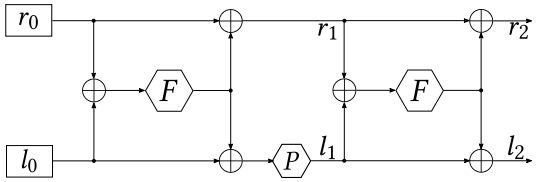
Step 3: Completeness. To establish the other implication $(ii) \Rightarrow (i)$, we have to show that if a constraint system Γ has no solutions, then no simulators can satisfy its constraints with overwhelming probability. However, impossibility results seem to suggest the contrary [Unr10]: one typical problem is the blowup of the number of collisions when xoring increasingly many random group elements. These collisions are not captured by symbolic models, and become overwhelmingly probable when performing a number of xors that is proportional to the security parameter η. Our proof therefore amounts to combinatorial arguments relying on the fact that our grammar only allows to specify distinguishers that perform a constant number of operations. We show that this restriction is necessary in Appendix 6.2, i.e., we construct an unsound example in an extension of our syntax with parametric loops.

4.4 Constraint solving

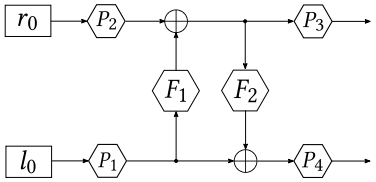
The main theorem reduces the decision of universality to a combination of symbolic notions (constraint systems and static equivalence): the natural next step is to search for decidability results for them. They are indeed standard, at least in the context of security protocols, and have received in-depth academic scrutiny.

Constraint satisfiability. Constraint systems have been studied in [Bau07, DLLT08, CR10, CCLD11, DKP12, CCD13, CKR18]. However, in $(\Phi, \mathsf{D}, \varphi)$, our generic grammar for φ is very permissive compared to the related work that typically only allows for conjunctions of (dis)equations; our prototype implementation therefore operates in a more restricted setting fitting the decidability results from the literature. It is known for example that the satisfiability problem ("has one constraint system a solution?") is decidable for *subterm convergent theories* [Bau07] (a class of theories including our equational theory for inverse cancellation) and *group theories* [DLLT08, DKP12] (which is more general than our theory of XOR). There then exists some combination results to obtain decidability in the union of the two disjoint theories [CR10], under some technical assumptions. However, the problem we study in the context of branching distinguishers (common solutions to several constraint systems) is less standard. In many cases though, the simulator constraints we obtain are simple enough to reduce the problem to a single-constraint setting (our prototype only answers when it manages to do so). Still, for future perspectives, one may note that our non-standard problem can be encoded as an *equivalence* in some cases. For example $(\Phi_1, \mathsf{D}, \varphi_1)$ and $(\Phi_2, \mathsf{D}, \varphi_2)$ have no common solutions *iff* $(\Phi_1, \mathsf{D}, \varphi_1)$ and $(\Phi_2, \mathsf{D}, \neg\varphi_2)$ have the same set of solutions. This equivalence of constraint systems has typically been studied in the context of subterm convergent [Bau07, CCLD11, CKR18] and group theories [DKP12], although we are not aware of combination results unlike in the case of satisfiability.

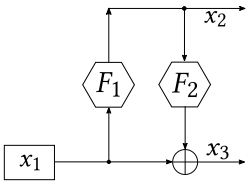
Static equivalence. The decidability of static equivalence has also been extensively studied [AC06, ACD07, BCLD07, CD12, CDK12]. As in the case of constraint systems, the problem is decidable for subterm-convergent theories [AC06, CDK12] (and is even NP-complete [CKR18]), for XOR [CD12], and combination results allow to obtain decidability for the union of disjoint theories [ACD07, CD12]. It is however important to remark that the cited references study the problem of static equivalence for *ground* frames (in our



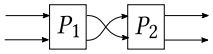
(a) 2-round Lai-Massey construction



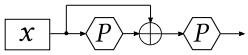
(b) 2-round Feistel sandwich of permutations



(c) 2-round unbalanced Feistel injection



(d) 2-round Confusion-Diffusion network



(e) Single-Permutation Davies-Meyer construction

Figure 7: Schematic description of some constructions on which we evaluate our prototype

context, this corresponds to the case where the responses of the simulator are never injected in subsequent distinguisher queries). A typical example where this is insufficient is the universal attack on 4-round Even-Mansour ciphers of [DSST17], involving a complex consistency argument that we were therefore not able to handle with our prototype (see Section 5). One may again note that, as in the case of satisfiability, the question of static equivalence with variables reduces to classical notions of process equivalence that are for example studied in [Bau07, CCLD11, CKR18].

In practice. All in all, our main theorem reduces universality to symbolic notions that are studied in the literature, althought not always under the exact same assumptions. However, the profusion of results still allowed us, with limited design effort, to cover many examples with our prototype AutoDiff by reducing the problems to settings where decidability is known. Our prototype proceeds this way, and issues a warning when it strays away from the cases where its correctness is established. We leave to future work the design of a complete decision procedure for our notion of constraint solving to obtain the strict decidability of distinguisher's universality.

5 PROTOTYPE AND AUTOMATED ATTACKS

Based on the presented approach so far, we implement AutoDiff, an automated tool for indifferentiability attacks with two modes:

- *verification*: on input of a candidate distinguisher code c, AutoDiff applies our main theorem to check whether $[\![c]\!]$ is universal.
- *synthesis*: AutoDiff attempts to generate a candidate universal distinguisher that is then tested as in verification mode. This is a heuristics: it does not guarantee that, if C is universally differentiable, a universal distinguisher will effectively be found.

We stress again that the verification of fixed attack codes (Verification Mode) is non-trivial, due to the universal quantification over all simulators inherent to the definition of indifferentiability. In both modes, the users specifies the construction C in the reference theory; the inverse C^{-1} should also be specified in case C is invertible, but the tool checks for safety that they are effectively inverses one from each other by solving a unification problem.

We evaluate the verification mode by formalising and corroborating existing indifferentiability attacks from the literature, while the synthesis mode is tasked to rediscover indifferentiability attacks without guidance. The benchmarks include constructions presented previously in our paper (Feistel networks, Iterated Even-Mansour ciphers) and some others presented in this section, among others in Figures 7a to 7e. All the experiments were executed on an 8-core machine with 1.80GHz Intel Core i7-10510U CPU and 16GB of RAM. The code of the tool is open-source (temp. review link):

https://www.dropbox.com/s/1imnc8c0eopv4gz/autodiff.zip?dl=0

5.1 Verifying attacks

We verify existing attacks from the literature as well as negative examples (invalid attacks) in Table 1. The approach is compatible with attacks under weaker security notions, e.g., the IND-CCA attack on 3-round Feistel [BF15] or the IND-CPA attack on palindromic Feistel networks (that are Feistel networks where the round functions $F_1, \ldots, F_k$ form a palindrome, i.e., $F_1 = F_k$, $F_2 = F_{k-1}$, and so on). It is known that these networks suffer from many vulnerabilities, in particular using the following (universal) distinguisher code D_{Fpal}, with $r_0, r_1 \in \mathsf{Rnd}$ two distinct random group elements:

1 $x_0, x_1 \leftarrow \mathbf{C}(r_0, r_1)$;
2 $y_1, y_0 \leftarrow \mathbf{C}(x_1, x_0)$;
3 return $r_0 =^? y_0 \wedge r_1 =^? y_1$

We note that the analysis of the attack on 4-round IEM [DSST17], our tool threw a warning stating that the underlying static equivalence problem contains variables and, therefore, is out of the scope where our procedure is sound (as mentioned in Section 4.4).

5.2 Synthesising attacks

We now present two fully automated heuristics that only take the construction as an input to derive a universal distinguisher against it. The naive approach would be to try all distinguisher codes by

Table 1: Performances of AutoDiff (Verification Mode)

Attack type	Construction	#Rounds	Attack reference	Result
IND-CPA IND-CCA	Palindromic Feistel network (PFN)	$\leqslant 10$	Distinguisher D_{Fpal}	✓ < 1 ms
Indifferentiability	Feistel Network (FN)	3	Barbosa and Farshim [BF15, Section 7]	✓ < 1 ms
		3	Figure 6 (Daisy)	↯ < 1 ms
		3	Figure 6 (David)	✓ < 1 ms
		5	Coron et al. [CHK+16, Section 2]	✓ 8 ms
		6	Greedy attack	↯ 11 ms
	Iterated Even-Mansour (IEM)	3	Lampe and Seurin [LS13, Section 3.2]	✓ 6 ms
		4	Dai et al. [DSST17, Section 3]	✗ 165 s
	Unbalanced Feistel injection	2	Barbosa and Farshim [BF18, Figure 8]	✓ < 1 ms
	Merkle-Damgård	—	Kelsey and Kohno [KK06]	✓ < 1 ms
	Confusion-Diffusion Network (CDN)	2	Dodis et al. [DSSL16, Section 3]	✓ 2 ms

✓ Universality proved ↯ Universality disproved ✗ Unable to conclude

increasing size and leverage the verification mode to discard invalid attacks—this is however intractable in practice. Instead, we developed two generic heuristics that exploit the internal structure of the construction C to synthesise more promising candidate distinguishers. Intuitively, they exploit the fact that the definition of C has some structure—at least more than an ideal function—and therefore verifies non-trivial identities. They propose two different approaches for deriving such relations, and then synthesise a distinguisher accordingly. We give more details below, using the same notations as in the reference theory, for a construction $C^{\mathcal{Q}} : G^{p_{in}} \to G^{p_{out}}$.

Heuristic 1: Relations first. This heuristic tries to find a pair of binary relations φ_1, φ_2 such that, in the real world, inputs $\mathbf{x}_1, \mathbf{x}_2$ such that $\mathbf{x}_1 \varphi_1 \mathbf{x}_2$ get mapped into outputs such that $C(\mathbf{x_1}) \varphi_2 C(\mathbf{x_2})$, expecting that it will be hard for the simulator to replicate this behaviour using only the ideal function R. For that we first consider sequences of variables $(\mathbf{x}_i : G^{p_{in}}, \mathbf{y}_i : G^{p_{out}})$ where $i \in [1, \ell]$ for some $\ell \in \mathbb{N}$. Intuitively, each $\mathbf{y}_i$ represents the output of C for the input $\mathbf{x}_i$. The heuristic then generates a system of equations as follows. It generates two arbitrary terms $\varphi_1(\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$ and $\varphi_2(\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$. For each pair of such terms, our algorithm considers:

$$\varphi_1(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) \stackrel{?}{=} 0 \wedge \varphi_2(\mathbf{y}_1, \ldots, \mathbf{y}_\ell) \stackrel{?}{=} 0 \wedge \bigwedge_{i=1}^{\ell} \mathbf{y}_i \stackrel{?}{=} C(\mathbf{x}_i)$$

with, in addition, some extra disequality constraints to guarantee that φ_1 or φ_2 would not be zero in the simulated world.

The described system is then solved using unification modulo theory, that is, we instantiate the variables of the system by ground terms of $\mathcal{T}(\mathcal{F} \cup \mathsf{Rnd})$ in a way the constraints are satisfied. This can be used to build a distinguisher, that performs the necessary oracle calls to compute the instantiated terms. As several (partial) orders of queries may be possible, we found it relevant to try out the codes corresponding to each possible permutations; we have however not observed in practice significant blow-ups, since there are still some partial dependencies in the order that often reduce the possibilities. Finally, the distinguisher returns φ_2 as its final test. The heuristics then submits the distinguisher to verification; in case of failure, it tries the procedure again with other terms φ_1 and φ_2, until all terms of a given size s have been tried ($s = 7$ in Table 2).

Heuristic 2: Expressions first. This heuristic tries, instead, to find a linear combination of expressions that is symbolically equal to 0 in the real world, but in a non-trivial manner so that it is hard for the simulator to replicate this behaviour. For some number arbitrary parameters d (the results of Table 2 are for $d = 5$), it generates all terms of $\mathcal{T}(\mathsf{Orcl} \cup \mathsf{Rnd})$ of size d or less, up to bijective renaming of random group elements. It then starts testing $\oplus$-combinations of some of these terms, until finding a combination e that is non-null in general, but null after replacing each occurrence of $\mathbf{C}(\mathbf{t})$ by the real-world definition $C(\mathbf{t})$. Intuitively, answering queries in a way that e evaluates to 0 in the simulated world is expectedly hard. As in the first heuristics, distinguishers are then derived from e by making the oracle queries necessary to compute it, followed by return $e \stackrel{?}{=} 0$. The process keeps going until a universal distinguisher is found or all possible linear combinations have been exhausted.

Table 2: Performances of AutoDiff (Synthesis Mode)

Construction	#Rounds	Attack type	Heuristics #1	Heuristics #2
FN	2	IND-CPA	✓ < 1 s	✓ < 1 s
	3	IND-CCA	✓ < 1 s	✓ < 1 s
	4	Indiff.	✓ 3 s	⏱
	5		✓ 53 s	⏱
	6		✗ 73 s	⏱
PFN	2	IND-CPA	✓ < 1 s	✓ < 1 s
	3	IND-CCA	✓ < 1 s	✓ < 1 s
	4		✓ 2 s	✓ < 1 s
	5		✓ 67 s	✓ < 1 s
	6		✗ 75 s	✓ < 1 s
IEM	1	IND-CPA	✗ < 1 s	✓ < 1 s
	2	Indiff.	✗ < 1 s	✓ 50 s
	3		✗ 1 s	⏱
LM (Fig.7a)	1	Indiff.	✓ < 1 s	✓ < 1 s
	2		✓ 21 s	⏱
FPS (Fig.7b)	2	Indiff.	✓ 2 s	⏱
UFI (Fig.7c)	2		✗ < 1 s	✓ < 1 s
CDN (Fig.7d)	2		✓ < 1 s	✓ 10 s
DMC (Fig.7e)	—		✗ < 1 s	✓ 16 s

✓ Attack found ✗ No attack found ⏱ Timeout (>100 seconds)

Results. Our results on attack synthesis are gathered in Table 2. An attack that does not require to invoke the round functions is

classified as IND-CCA, and as IND-CPA if it only uses the main oracle and not its inverse. We point out that, as the above description suggests, the heuristics are defined with a general purpose and are not specifically tailored to our set of benchmarks. Also observe that our heuristics complement each other well in that an attack is found for most of the case studies by at least one of them. Heuristic 1 remarkably synthesises a universal attack against 5-round Feistel network fully automatically. On the other hand, none of our heuristics could attack the iterated Even-Mansour construction with more than 2 rounds. We expect that improving our heuristics by making use of branching may lead to covering these missing cases.

6 DISCUSSION AND EXTENSIONS

We discuss here some side results and related research directions, as well as technical limits and improvements of our contributions.

6.1 Collision finding

We also experimented our approach in other domains of cryptography, for instance, finding collisions to hash functions. We reduce this problem to a similar (yet much more elementary) notion of constraint solving that can be analysed by a fragment of AutoDiff (or most other general-purpose symbolic tools). As an illustrative example, consider the 64 basic ways to construct a hash function $H : \{0,1\}^* \to \{0,1\}^\eta$ from a block cipher $E : \{0,1\}^\eta \times \{0,1\}^\eta \to \{0,1\}^\eta$, the *PGV functions*, proposed by Preneel et al. [PGV94]. These were classified, according to different levels of security, into several groups, which were later refined [BRS02], one of the relevant studied properties being *collision resistance*. It was concluded that only 12 of them satisfied strong notions of security.

We express collisions for these hash function candidates in a natural way, i.e., as equality constraints. In these terms, collision resistance therefore rephrases to *unification* in the reference theory, that is, the question of finding, given two terms u, v, a substitution σ such that $u\sigma =_{\mathcal{E}} v\sigma$, if any. We could then leverage the constraint solving algorithms developed in AutoDiff; this resulted in AutoDiff automatically identifying the same set of 12 distinguished functions.

6.2 Assumption of boundedness

Algebraic distinguishers have two main restrictions:

(1) they only perform operations from the reference theory, and
(2) the number of operations they perform is fixed, i.e., it does not increase with the security parameter η.

The first restriction is the reason we are able to carry out a symbolic analysis at all. One could however wonder whether one could get rid of the second one, for example, by allowing distinguisher codes to perform loops of length η. We can actually show that this assumption is necessary; by that we mean that there exists a distinguisher D, performing a number of operations proportional to η, against a construction C that will be considered as universal by our main theorem while there exists a simulator S winning against D. This relies on the following simple observation, recalling similar impossibility results in security-protocol analysis [Unr10]:

Proposition 6.1. *Let E be a set of $\eta + 1$ pairwise distinct η-bitstrings. Then there exists a non-empty subset $P \subseteq E$, efficiently computable from E, such that $\sum_{x\in P} x = 0$ (where $\sum$ means xoring).*

Proof. It suffices to observe that there exists $2^{\eta+1}$ subsets of E but only 2^η different bitstrings; by the pigeonhole principle, there needs to exist $P_1, P_2 \subseteq E$ distinct such that $\sum_{x\in P_1} x = \sum_{x\in P_2} x$. By the self-cancellation property of xor, we therefore obtain the expected conclusion by taking the symmetric difference $P = P_1 \Delta P_2$. The set P can then easily be computed from E by Gaussian elimination, seeing G as the vector space $\mathrm{GF}(2)^\eta$. □

This shows that our symbolic model, that does not capture any algebraic relations between random group elements, is unsound if there are $\eta + 1$ of them. In the bounded case, our proof of the main theorem precisely relies on the fact that the probability that a non-empty subset $P \subseteq E$ of null sum exists is negligible if E contains a bounded number of uniformly sampled bitstrings (rather than $\eta + 1$). But let us construct a concrete counterexample, given the construction $C(x) = x$ and a primitive $F : G \to G$. Consider then the following (unbounded) distinguisher code:

```
1  x_0 ← F(r_0); ...; x_η ← F(r_η);
2  x ← C(r);
3  return (x_0, ..., x_η) ≠? (0, ..., 0) ∧ (x = r ∨ φ)
```

with random group elements $r, r_0, \ldots, r_\eta$, and where the formula

$$\varphi = \bigwedge_{i=0}^{\eta} (x_i =^? 0 \vee x_i =^? r_i) \wedge \sum_{i=0}^{\eta} x_i =^? 0$$

models that the set $\{x_0, \ldots, x_\eta\} \setminus \{0\}$ is a subset of $\{r_0, \ldots, r_\eta\}$ of null sum. Intuitively, the distinguisher sends $\eta + 1$ random values r_i to the simulator, and then checks that either (1) $\mathbf{C}(r) = r$, which holds with overwhelming probability *iff* the distinguisher is executed in the real world; (2) or that φ holds. The constraint system corresponding to this distinguisher has no solutions, but by Proposition 6.1, there exists a simulator satisfying this requirement.

Note that this example uses the expressivity of grammar for return formulas to its fullest (in particular, disjunctions), unlike most decidability results on constraint systems. The example is however not an artifact of our permissive formalism: provided we extend the reference theory with free symbols $\mathsf{h} \in \mathsf{Fun}$ (intuitively modelling publicly available random functions), the above return statement can be encoded with formulas that only allows conjunctions of equations $u =^? v$ and disequations $u \neq^? v$ between terms of arity 1. We do not detail the full encoding, but mention for example:

- $(u, u') \neq^? (v, v')$ can be encoded by $u \oplus \mathsf{h}(u') \neq^? v \oplus \mathsf{h}(v')$, thus allowing to model comparisons of tuples;
- $x_i =^? 0 \vee x_i =^? r_i$ by $\mathsf{h}(0) \oplus \mathsf{h}(x_i) \oplus \mathsf{h}(r_i) \oplus \mathsf{h}(x_i \oplus r_i) =^? 0$, thus allowing to model most disjunctions of interest.

6.3 Extending the reference theory

In this paper we limited ourselves to a reference theory modelling xor and inverse-cancellation. But from a decidability standpoint, the settings where constraint satisfiability is known to be decidable are usually stable by addition of free function symbols. They would typically model public permutations as in general confusion-diffusion mechanisms, or to some extent in Substitution-Permutation networks such as AES. More generally, our reference theory is rather restricted compared to the amount of theories that have been studied in unification theory, which suggests that scope improvements of our approach are possible to obtain with reasonable effort.

6.4 Proofs of universality

A natural follow-up question is whether our approach may *prove* indifferentiability, i.e., check the validity of a given simulator. Actually, simulators may be specified in calculi of communicating processes such as the applied pi calculus [ABF17]. Consider $C(x) = x \oplus F(x)$ for example, and a simulator S^R that emulates queries to $F(x)$ by returning $R(x) \oplus x$. In the applied pi calculus, the real and simulated oracle systems can be modelled by the processes P_{real} and P_{sim}:

$$P_{real} = !\,\mathsf{in}(q_c, x); \mathsf{out}(q_c, x \oplus F(x)) \;\mid !\,\mathsf{in}(q_F, x); \mathsf{out}(q_F, F(x))$$
$$P_{sim} = !\,\mathsf{in}(q_c, x); \mathsf{out}(q_c, R(x)) \;\mid !\,\mathsf{in}(q_F, x); \mathsf{out}(q_F, R(x) \oplus x)$$

These processes model an unbounded number of queries to (1) the main oracle **C**, under the form of an input from the adversary on a channel q_c, written $\mathsf{in}(q_c, x)$, answered by output $u \in \{C(x), R(x)\}$, written $\mathsf{out}(q_c, u)$; and to (2) the primitive F, similarly. An interesting lead is whether the validity of S can be reduced to the decision of processes equivalences [ABF17, CKR18], e.g., *trace equivalence*, modelling the indistinguish distinguish between the two processes.

7 CONCLUSION

We have developed and implemented symbolic methods for synthesising universal distinguishers against indifferentiability. Our method covers a broad set of examples from the literature, including Feistel networks and Iterated Even-Mansour blockciphers. Interesting directions for future work include decidability of constraint solving, and support for public permutations in our syntax (e.g., for larger Confusion-Diffusion networks, or Substitution-Permutation networks), or adapting the approach to *prove* indifferentiability.

REFERENCES

[ABF17] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM (JACM)*, 2017.

[ABGW17] Miguel Ambrona, Gilles Barthe, Romain Gay, and Hoeteck Wee. Attribute-based encryption in the generic group model: Automated proofs and new constructions. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 647–664, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[ABS16] Miguel Ambrona, Gilles Barthe, and Benedikt Schmidt. Automated unbounded analysis of cryptographic constructions in the generic group model. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 822–851, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[AC06] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1-2):2–32, 2006.

[ACD07] Mathilde Arnaud, Véronique Cortier, and Stéphanie Delaune. Combining algorithms for deciding knowledge in security protocols. In Boris Konev and Frank Wolter, editors, *Frontiers of Combining Systems*, pages 103–117, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[AR07] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptol.*, 20(3):395, 2007.

[Bau07] Mathieu Baudet. *Sécurité des protocoles cryptographiques: aspects logiques et calculatoires.* PhD thesis, École normale supérieure de Cachan, 2007.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

[BBK17] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the tls 1.3 standard candidate. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[BCG+13] Gilles Barthe, Juan Manuel Crespo, Benjamin Grégoire, César Kunz, Yassine Lakhnech, Benedikt Schmidt, and Santiago Zanella Béguelin. Fully automated analysis of padding-based encryption in the computational model. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 1247–1260, Berlin, Germany, November 4–8, 2013. ACM Press.

[BCK09] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. *Information and Computation*, 207(4):496–520, 2009.

[BCLD07] Sergiu Bursuc, Hubert Comon-Lundh, and Stéphanie Delaune. Associative-commutative deducibility constraints. In Wolfgang Thomas and Pascal Weil, editors, *STACS 2007*, pages 634–645, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[BDF11] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 169–187, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[BDH+18] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5g authentication. In *ACM Conference on Computer and Communications Security (CCS)*, 2018.

[BDPVA08] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EuroCrypt)*, 2008.

[BF15] Manuel Barbosa and Pooya Farshim. The related-key analysis of Feistel constructions. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption – FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 265–284, London, UK, March 3–5, 2015. Springer, Heidelberg, Germany.

[BF18] Manuel Barbosa and Pooya Farshim. Indifferentiable authenticated encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 187–220, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[BFF+14] Gilles Barthe, Edvard Fagerholm, Dario Fiore, John C. Mitchell, Andre Scedrov, and Benedikt Schmidt. Automated analysis of cryptographic assumptions in generic group models. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 95–112, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[BFG+18] Gilles Barthe, Xiong Fan, Joshua Gancher, Benjamin Grégoire, Charlie Jacomme, and Elaine Shi. Symbolic proofs for lattice-based cryptography. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 538–555, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[BGJ+19] Gilles Barthe, Benjamin Grégoire, Charlie Jacomme, Steve Kremer, and Pierre-Yves Strub. Symbolic methods in computational cryptography proofs. In *31st IEEE Computer Security Foundations Symposium (CSF)*, 2019.

[BGS15] Gilles Barthe, Benjamin Grégoire, and Benedikt Schmidt. Automated proofs of pairing-based cryptography. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1156–1168, Denver, CO, USA, October 12–16, 2015. ACM Press.

[BPW06] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. A zero-dimensional Gröbner basis for AES-128. In Matthew J. B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 78–88, Graz, Austria, March 15–17, 2006. Springer, Heidelberg, Germany.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Workshop on the Theory and Application of of Cryptographic Techniques*, 1994.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[CCD13] Vincent Cheval, Véronique Cortier, and Stéphanie Delaune. Deciding equivalence-based properties using constraint solving. *Theoretical Computer Science*, 2013.

[CCLD11] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Trace equivalence decision: Negative tests and non-determinism. In *ACM conference on Computer and communications security (CCS*, 2011.

[CD12] Véronique Cortier and Stéphanie Delaune. Decidability and combination results for two notions of knowledge in security protocols. *Journal of Automated Reasoning*, 48(4):441–487, 2012.

[CDE+08] Judicaël Courant, Marion Daubignard, Cristian Ene, Pascal Lafourcade, and Yassine Lakhnech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 371–380, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press.

[CDK12] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 2012.

[CHH+17] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of tls 1.3. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[CHK+16] Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the feistel construction. *Journal of Cryptology*, 29(1):61–114, Jan 2016.

[CKR18] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. DEEPSEC: Deciding equivalence properties in security protocols theory and practice. In *IEEE Symposium on Security and Privacy (S&P)*, 2018.

[CKW11] Véronique Cortier, Steve Kremer, and Bogdan Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *Journal of Automated Reasoning*, 2011.

[CLC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In *ACM conference on Computer and communications security (CCS)*, 2008.

[CM03] Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[Cou03] Nicolas Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[CR10] Yannick Chevalier and Michael Rusinowitch. Symbolic protocol analysis in the union of disjoint intruder theories: Combining decision procedures. *Theoretical Computer Science*, 2010.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 157–184, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[DKP12] Stéphanie Delaune, Steve Kremer, and Daniel Pasaila. Security protocols, constraint systems, and group theories. In *International Joint Conference on Automated Reasoning (IJCAR)*, 2012.

[DLLT08] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 2008.

[DS16] Yuanxi Dai and John P. Steinberger. Indifferentiability of 8-round Feistel networks. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 95–120, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[DSSL16] Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of confusion-diffusion networks. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 679–704, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[DSST17] Yuanxi Dai, Yannick Seurin, John P. Steinberger, and Aishwarya Thiruvengadam. Indifferentiability of iterated Even-Mansour ciphers with non-idealized key-schedules: Five rounds are necessary and sufficient. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 524–555, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[DY81] D. Dolev and A.C. Yao. On the security of public key protocols. In *Symposium on Foundations of Computer Science (FOCS)*, 1981.

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of cryptology*, 1997.

[Fei73] H. Feistel. *Cryptography and Computer Privacy*. Scientific American, 1973.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. In *Annual International Cryptology Conference (CRYPTO)*, 2001.

[HKM15] Viet Tung Hoang, Jonathan Katz, and Alex J. Malozemoff. Automated analysis and synthesis of authenticated encryption schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 84–95, Denver, CO, USA, October 12–16, 2015. ACM Press.

[HKT10] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. Equivalence of the random oracle model and the ideal cipher model, revisited. *CoRR*, abs/1011.1264, 2010.

[JK21] Charlie Jacomme and Steve Kremer. An extensive formal analysis of multi-factor authentication protocols. *ACM Transactions on Privacy and Security (TOPS)*, 2021.

[KK06] John Kelsey and Tadayoshi Kohno. Herding hash functions and the nostradamus attack. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2006.

[KM07] Steve Kremer and Laurent Mazaré. Adaptive soundness of static equivalence. In *European Symposium on Research in Computer Security (ESORICS*, 2007.

[Leu12] Gaëtan Leurent. Analysis of differential attacks in ARX constructions. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 226–243, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[LS13] Rodolphe Lampe and Yannick Seurin. How to construct an ideal cipher from a small set of public permutations. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 444–463, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12, Cirencester, UK, December 19–21, 2005. Springer, Heidelberg, Germany.

[Mea20] Catherine Meadows. Symbolic and computational reasoning about cryptographic modes of operation. Cryptology ePrint Archive, Report 2020/142, 2020. https://eprint.iacr.org/2020/794.

[MKG14] Alex J. Malozemoff, Jonathan Katz, and Matthew D. Green. Automated analysis and synthesis of block-cipher modes of operation. In Anupam Datta and Cedric Fournet, editors, *CSF 2014: IEEE 27st Computer Security Foundations Symposium*, pages 140–152, Vienna, Austria, jul 19-22 2014. IEEE Computer Society Press.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[PGV94] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[SSD+18] Danping Shi, Siwei Sun, Patrick Derbez, Yosuke Todo, Bing Sun, and Lei Hu. Programming the Demirci-Selçuk meet-in-the-middle attack with constraints. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 3–34, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

[Unr10] Dominique Unruh. The impossibility of computationally sound xor. *IACR Cryptol. ePrint Arch.*, 2010.

A PROOF OF THE MAIN THEOREM

Main theorem. *Let $D = [\![c_1 + \cdots + c_n]\!]$ be an algebraic distinguisher for some non-branching distinguisher codes c_i. Then the following points are equivalent:*

(i) D is universal
(ii) the constraint systems $\Gamma(c_1), \ldots, \Gamma(c_n)$ have no common, consistent solutions Σ

A.1 Some notations

We interpret ground terms u and ground frames Φ as distributions of bitstrings $[\![u]\!]$ and distributions of *stores* $[\![\Phi]\!]$, respectively, where a store is a mapping from axioms to bitstrings. We interpret:

- random group elements as uniformly sampled bitstrings
- symbols of Fun in the natural way
- symbols of Orcl_0^* (resp. $\mathsf{Orcl}_0 \setminus \mathsf{Orcl}_0^*$) as function uniformly sampled from the set of all keyed permutations (resp. all functions) of adequate domain.

We also write $\Pr[\phi \leftarrow \mathcal{D} : E]$ to refer to the probability of the event E conditioned by drawing a store ϕ with respect to the distribution of stores $\mathcal{D}$. Finally, given two distributions on stores $\mathcal{D}_1$ and $\mathcal{D}_2$, we say that they are *indistinguishable*, written $\mathcal{D}_1 \approx \mathcal{D}_2$, when for all algorithms S from stores to $\{0,1\}$ running in polynomial time in the security parameter, we have that

$$\left|\Pr[\phi \leftarrow \mathcal{D}_1 : S(\phi) = 0] - \Pr[\phi \leftarrow \mathcal{D}_2 : S(\phi) = 0]\right|$$

is negligible. They are *strongly distinguishable* if there exists a polynomial algorithm S such that the above quantity is overwhelming.

A.2 First direction

Let us prove first the direction $(i) \Rightarrow (ii)$ of the theorem, by contraposition. Consider a distinguisher $D = [\![c_1 + \cdots + c_n]\!]$, with c_i non branching for all i, against a construction $C^{\mathcal{Q}}$ and assume that there exists a solution Σ to each $\Gamma(c_i)$ that satisfies the consistency requirement. Let us also call $\sigma_1, \ldots, \sigma_n$ the first-order solutions of Σ in the constraint systems $\Gamma(c_1), \ldots, \Gamma(c_n)$, respectively. Our goal is to construct a simulator that wins the security game against D with overwhelming probability. To obtain the desired conclusion, it suffices to prove the following points:

Proposition A.1. *Then we have:*

(1) if $u =_{\mathcal{E}} v$ then $\Pr[e, f \leftarrow [\![u, v]\!] : e = f]$ is overwhelming
(2) if $u \neq_{\mathcal{E}} v$ then $\Pr[e, f \leftarrow [\![u, v]\!] : e = f]$ is negligible
(3) if u is deducible from Φ, then there exists a polynomial time algorithm S from stores to bitstrings such that $\Pr[\phi, e \leftarrow [\![\Phi]\!], [\![u]\!] : S(\phi) = e]$ is overwhelming
(4) if $\Phi \sim \Psi$, then $[\![\Phi]\!] \approx [\![\Psi]\!]$
(5) if $\Phi \not\sim \Psi$, then $[\![\Phi]\!]$ and $[\![\Psi]\!]$ are strongly distinguishable.

Item (4) is not needed obtaining the conclusion, but rather as an intermediary argument. Once we obtain all these items, it suffices to instantiate all equality and disequality constraints in each c_i by σ_i; they will all be satisfied by Items (3) and (1), and the deducibility constraints as well by Item (2). Besides, in case some deducibility constraints are solved by different second-order terms, we know by consistency of Σ that the underlying frames are not statically equivalent. In particular by Item (5) there exists a polynomial algorithm for distinguishing the two simulator states with overwhelming probabilities, allowing to answer the two queries differently. Regarding the proof of these items, we have in the reference theory:

- (1) holds in a straightforward manner (the probability is even 1), as all equations of the reference theory hold with probability 1 for their bitstring interpretations;
- (1)⇒(3) by [BCK09, Proposition 1];
- (1∧2)⇒(5) by [BCK09, Proposition 1];
- (4)⇒(2) by [BCK09, Proposition 2].

It therefore only remains to prove Item (4). For that we use the following characterisation, corresponding to [BCK09, Proposition 3] and applicable to the reference theory by Item (1). The proposition makes reference, given a frame Φ, to its *ideal semantics* $[\![\Phi]\!]^{\mathsf{ideal}}$:

Proposition A.2. *If for every ground frame Φ, $[\![\Phi]\!] \approx [\![\Phi]\!]^{\mathsf{ideal}}$, then Item* (2) *of Proposition* (A.1) *holds.*

The precise definition of the distribution of stores $[\![\Phi]\!]^{\mathsf{ideal}}$ is not needed in our proofs: indeed, the theory of xor is specifically studied in [BCK09], and the following result is already established (as a combination of [BCK09, Theorem 6, Theorem 7, Proposition 8]):

Proposition A.3. *If Φ is a frame that is only constructed from random group elements, 0, and $\oplus$, then $[\![\Phi]\!] \approx [\![\Phi]\!]^{\mathsf{ideal}}$.*

To conclude the whole proof, it therefore suffices to prove that for all frames Φ in the reference theory, there exists a frame Ψ such that $[\![\Phi]\!] \approx [\![\Psi]\!]$ and Φ only uses $\oplus$ as a non-constant function symbol. For that it sufficies to design a procedure that removes gradually $\mathbf{C}$ and $\mathbf{C}^{-1}$ symbols from Φ while keeping the same frame distribution as the initial one. Since we consider ideal random functions, it suffices to replace each subterm of Φ rooted with $f \in \{\mathbf{C}, \mathbf{C}^{-1}\}$, by fresh random group elements, where $f(u)$ and $f(v)$ should be mapped to the same random group element when $u =_{\mathcal{E}} v$.

A.3 Converse direction

Now let $D = [\![c_1 + \cdots + c_n]\!]$ be a distinguisher, and S be a simulator winning against D with overwhelming probability. We want to construct a consistent solution to $\Gamma(c_1), \ldots, \Gamma(c_n)$. Ae can assume S to be deterministic: if there exists a convex combination of deterministic simulators winning with overwhelming probability $1 - \varepsilon(\eta)$ against D, then one of these simulators wins with probability greater or equal than $1 - \varepsilon(\eta)$ against D as well. Besides, we can also argue that it suffices to consider simulators performing a constant number of oracle queries. Indeed, up to the addition of dummy deducibility constraints and constant entries to the frame, we can always assume that each $\Gamma(c_i) = (\Phi, \mathsf{D}, \varphi)$ is of the form

$$\Phi = \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n\}$$
$$\mathsf{D} = \{X_1 : 1 \vdash^? x_1, \ldots, X_n : n \vdash^? x_n\} \ .$$

Consider a simulator S computing bitstrings $x_1, \ldots, x_n$ with access to the corresponding substores sampled from $[\![\Phi]\!]$. There are only $2^{|\varphi|+|t_1|}$ subterms of the system when the simulator computes x_1, therefore it has no effect on constraint satisfiability to perform oracle calls on other terms; all other calls $R(t)$ can be replaced by any bitstring without affecting constraint satisfiability. Assuming the maximum number of calls have been performed, and taking into account that t_2 may have been instanciated by a computation

σ, its size may be up to $|t_2\sigma| \leqslant e = |t_2|2^{|\varphi|+|t_1|}$. Applying the same reasoning as for the first computation, here is then at most $2^{2^{|\varphi|+|t_1|}+e}$ non-spurious oracle calls that may not be replacable by constants. Iterating this process by induction allows us to conclude that the simulator can be assumed to perform a bounded number of oracle calls without loss of generality. In particular, writing A_i the algorithm derived from S to compute x_i, the determinism and constant number of oracle calls show that A_i only samples a fixed number of random values (one for each oracle call). Besides:

Proposition A.4. *Let $x_1, \ldots, x_n$ be a constant number of values sampled independently uniformly at random. Then, with overwhelming probability, the only subset $P \subseteq \{x_1, \ldots, x_n\}$ of null sum is $P = \emptyset$.*

Proof. The proof can easily be obtained by recurrence as, assuming $x_1, \ldots, x_{n-1}$ verify this property, the probability of a collision (that is, that $x_n = \sum_{x \in P} x$ for some subset $P \subseteq \{x_1, \ldots, x_{n-1}\}$) is bounded by $2^{n-\eta}$, which is negligible due to n being constant. □

In particular, combining this result with the abovementioned conclusion that the number of samplings of S is constant, we obtain that there is a negligible probability of collision between any linear combination of computations of the simulators, unless the collision holds with probability 1—which exactly means that a symbolic solution verifying the same equalities as S can be derived. We therefore obtain the expected conclusion, which concludes the proof.