Public-Seed Pseudorandom Permutations

Pratik Soni and Stefano Tessaro

University of California, Santa Barbara
{pratik_soni,tessaro}@cs.ucsb.edu

Abstract. This paper initiates the study of standard-model assumptions on permutations – or more precisely, on families of permutations indexed by a *public* seed. We introduce and study the notion of a *public-seed pseudorandom permutation* (psPRP), which is inspired by the UCE notion by Bellare, Hoang, and Keelveedhi (CRYPTO '13). It considers a two-stage security game, where the first-stage adversary is known as the source, and is restricted to prevent trivial attacks – the security notion is consequently parameterized by the class of allowable sources. To this end, we define in particular unpredictable and reset-secure sources analogous to similar notions for UCEs.

We first study the relationship between psPRPs and UCEs. To start with, we provide efficient constructions of UCEs from psPRPs for both reset-secure and unpredictable sources, thus showing that most applications of the UCE framework admit instantiations from psPRPs. We also show a converse of this statement, namely that the five-round Feistel construction yields a psPRP for reset-secure sources when the round function is built from UCEs for reset-secure sources, hence making psPRP and UCE equivalent notions for such sources.

In addition to studying such reductions, we suggest generic instantiations of psPRPs from both block ciphers and (keyless) permutations, and analyze them in ideal models. Also, as an application of our notions, we show that a simple modification of a recent highly-efficient garbling scheme by Bellare et al. (S&P '13) is secure under our psPRP assumption.

Keywords: Symmetric cryptography, UCE, permutation-based cryptography, assumptions, indifferentiability

1 Introduction

Many symmetric cryptographic schemes are built generically from an underlying component, like a hash function or a block cipher. For several recent examples (e.g., hash functions [15,39], authenticated-encryption schemes [4], PRNGs [16], garbling schemes [10]), this component is a (keyless) *permutation*, which is either designed from scratch to meet certain cryptanalytic goals (as in the case of SHA-3 and derived algorithms based on the sponge paradigm) or is instantiated by fixing the key in a block cipher like AES (as in the garbling scheme of [10]).

The security of these schemes is usually proved in the *ideal-permutation model*, that is, the permutation is randomly chosen, and all parties are given

(black-box) access to it. Essentially no non-tautological assumptions on permutations are known which are sufficient to imply security.[1] This situation is in sharp contrast to that of hash functions, where despite the popularity of the random-oracle model, we have a good understanding of plausible security assumptions that can be satisfied by these functions. This is particularly important – not so much because we want to put ideal models in question, but because we would like to assess what is really expected from a good permutation or hash function that makes these schemes secure.

Our contributions, in a nutshell. This paper initiates the study of computational assumptions for permutation-based cryptography. Akin to the case of hash functions, we extend permutations with a *public* seed, that is, π_s is used in lieu of π, where s is a public parameter of the scheme. We introduce a new framework – which we call *public-seed pseudorandom permutations*, or psPRPs, for short – which we investigate, both in terms of realizability, as well as in terms of applications. Our approach takes inspiration from Bellare, Hoang, and Keelveedhi's UCE framework [8], which we extend to permutations. As we will see, psPRPs are both useful and interesting objects of study.

Beyond definitions, we contribute in several ways. First off, we build UCEs from psPRPs via efficient permutation-based hash functions, and show conversely how to build psPRPs from UCEs using the Feistel construction. We also discuss generic instantiations of psPRPs from block ciphers and keyless permutations. Finally, we show a variant of the garbling scheme from [10] whose security can be based on a psPRP assumption on the underlying block cipher, without compromising efficiency. Our reductions between psPRPs and UCEs are established by general theorems that connect them with a weak notion of indifferentiability, which is of independent interest. We explain all of this in detail in the remainder of this introduction; an overview of the results is in Fig. 1.

The UCE framework: A primer. Bellare, Hoang, and Keelveedhi (BHK) [8] introduced the notion of a *universal computational extractor* (UCE). For a seeded hash function $\mathsf{H} : \{0,1\}^s \times \{0,1\}^* \to \{0,1\}^h$, the UCE framework considers a two-stage security game. First, a *source* S is given oracle access to either $\mathsf{H}(s,\cdot)$ (for a random, and for now secret, seed s), or a random function $\rho : \{0,1\}^* \to \{0,1\}^h$. After a number of queries, the source produces some leakage $L \in \{0,1\}^*$. In the second stage, the distinguisher D learns *both* L and the seed s, and needs to decide whether S was interacting with $\mathsf{H}(s,\cdot)$ or ρ – a task we would like to be hard. Clearly, this is unachievable without restrictions on S, as it can simply set $L = y^*$, where y^* is the output of the oracle on a fixed input x^*, and D then checks whether $\mathsf{H}(s,x^*) = y^*$, or not.

BHK propose to restrict the set of allowable sources – the security notion $\mathsf{UCE}[\mathcal{S}]$ corresponds to a function H being secure against all sources within a class $\mathcal{S}$. For example, *unpredictable* sources are those for which a predictor P,

[1] A notable exception is the line of work on establishing good bounds on the PRF-security of MACs derived from sponge-based constructions, as e.g. in [37,3,26], where one essentially assumes that the underlying permutation yields a secure Even-Mansour [25] cipher.

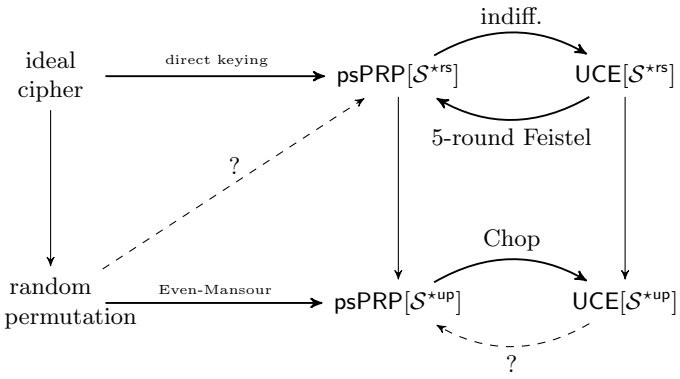

Fig. 1: **Relations established in this paper.** Here, $\star$ is set consistently everywhere either to c or to s. Lack of arrow indicates a separation, dashed lines indicate implications that are open and which we conjecture to hold true. Also note that in the ideal-cipher model, a random permutation is obtained by fixing the cipher key (e.g., to the all-zero string). We do not know whether the converse is true generically – indifferentiable constructions of ideal ciphers from random permutations (e.g., [1]) do not apply here [38].

given the leakage $L \xleftarrow{\$} S^{\rho}$, cannot guess any of S's queries. They further distinguish between the class of *computationally* unpredictable sources $\mathcal{S}^{\mathsf{cup}}$ and the class of *statistically* unpredictable sources $\mathcal{S}^{\mathsf{sup}}$, depending on the powers of P. A somewhat stronger notion – referred to as *reset-security* – demands that a distinguisher R given $L \xleftarrow{\$} S^{\rho}$ accessing the random function ρ cannot tell whether it is given access to the same oracle ρ, or to a completely independent random oracle ρ'. One denotes as $\mathcal{S}^{\mathsf{srs}}$ and $\mathcal{S}^{\mathsf{crs}}$ the classes of statistical and computational reset-secure sources, respectively.

While $\mathsf{UCE}[\mathcal{S}^{\mathsf{cup}}]$-security (even under meaningful restrictions) was shown impossible to achieve in the standard model [17,14] assuming indistinguishability obfuscation (IO) [5], there is no evidence of impossibility for $\mathsf{UCE}[\mathcal{S}^{\mathsf{sup}}]$ and $\mathsf{UCE}[\mathcal{S}^{\mathsf{srs}}]$, and several applications follow from them. Examples include providing standard-model security for a number of schemes and applications previously only secure in the random-oracle model, including deterministic [8] and hedged PKE [7], immunizing backdoored PRGs [24], message-locked encryption [8], hardcore functions [8], point-function obfuscation [8,13] simple KDM-secure symmetric encryption [8], adaptively-secure garbling [8], and CCA-secure encryption [34]. Moreover, as also pointed out by Mittelbach [36], and already proved in the original BHK work, $\mathsf{UCE}[\mathcal{S}^{\mathsf{crs}}]$ and $\mathsf{UCE}[\mathcal{S}^{\mathsf{cup}}]$ are achievable in *ideal* models, and act as useful intermediate security notions for two-stage security games, where indifferentiability does not apply [38].

PUBLIC-SEED PRPS. We extend the UCE approach to the case of a *seeded* permutation $\pi : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^n$, that is, $\pi_s = \pi(s,\cdot)$ is an efficiently

invertible permutation on n-bit strings. As in the UCE case, the security game will involve a source making queries to a permutation P *and its inverse* P^{-1}. In the real case, P / P^{-1} give access to π_s and π_s^{-1}, whereas in the ideal case they give access to a random permutation ρ and its inverse ρ^{-1}. Then, S passes on the leakage L to the distinguisher D, which additionally learns s. The $\mathsf{psPRP}[\mathcal{S}]$ security notion demands indistinguishability of the real and ideal cases for all PPT D and all $S \in \mathcal{S}$.

This extension is straightforward, but it is not clear that it is useful at all. For instance, UCEs naturally generalize the notion of an extractor, yet no such natural "generalization" exists here, except that of extending the PRP game (played by the source) with a public-seed stage (and hence, the name psPRP). In addition, necessary source restrictions are somewhat less intuitive than in the UCE case. For instance, for psPRPs, for statistically/computationally unpredictable sources (we abuse notation, and denote the corresponding source classes also as $\mathcal{S}^{\mathsf{sup}}$ and $\mathcal{S}^{\mathsf{cup}}$) it must be hard for a predictor to guess an input *or* an output of the queries made by S.

UCEs from psPRPs. We first show that psPRPs are not only useful, but essentially allow to recover *all* previous applications of UCEs through simple constructions of UCEs from psPRPs.

Our first result shows that all permutation-based constructions which are *indifferentiable* from a random oracle [35,20] transform a $\mathsf{psPRP}[\mathcal{S}^{\star\mathsf{rs}}]$-secure permutation into $\mathsf{UCE}[\mathcal{S}^{\star\mathsf{rs}}]$-secure hash function, where $\star \in \{\mathsf{c}, \mathsf{s}\}$.[2] In particular, this implies that the sponge paradigm by Bertoni *et al* [15], which underlies the SHA-3 hash function, can be used for such transformation, thus giving extra validation for the SHA-3 standard. We note that the permutation underlying SHA-3 is not seeded, but under the assumption that the underlying permutation is $\mathsf{psPRP}[\mathcal{S}^{\star\mathsf{rs}}]$-secure when seeded via the Even-Mansour construction [25], it is easy to enhance the sponge construction with a seed.

Note that $\mathcal{S}^{\star\mathsf{rs}}$ is a strictly larger class than $\mathcal{S}^{\star\mathsf{up}}$ (for both psPRP and UCE). Therefore, when an application only needs $\mathsf{UCE}[\mathcal{S}^{\star\mathsf{up}}]$-secure hashing, one may ask whether the assumption on the underlying psPRP can also be reduced. We will prove that this is indeed the case, and show that whenever π is $\mathsf{psPRP}[\mathcal{S}^{\star\mathsf{up}}]$-secure, then the simple construction that on input X outputs $\pi_s(X)[1 \dots r]$, that is, the first r bits of $\pi_s(X)$ is a secure fixed-input length $\mathsf{UCE}[\mathcal{S}^{\star\mathsf{up}}]$ as long as $r < n - \omega(\log \lambda)$. This result can be combined with the domain extender of [9] to obtain a variable-input-length $\mathsf{UCE}[\mathcal{S}^{\star\mathsf{up}}]$-secure hash function.[3]

CP-sequential indifferentiability. The technique behind the above results is inspired by Bellare, Hoang, and Keelveedhi's work [9] on UCE domain extension. They show that every construction that transforms a fixed-input length

[2] We note that the computational case, by itself, is not that useful, given we know that $\mathsf{UCE}[\mathcal{S}^{\mathsf{cup}}]$ and hence also $\mathsf{UCE}[\mathcal{S}^{\mathsf{crs}}]$ security is unachievable, unless IO does not exist. However, we may want to occasionally apply these results in ideal models, where the notion is achievable, and thus they are worth stating.

[3] Their construction pre-processes the arbitrary-long input with an almost universal hash function, as e.g. one based on polynomial evaluation.

random oracle into a variable-input length one in the sense of indifferentiability [35,20] is a good domain extender for UCEs.

We extend their result along three axes. First off, we show that it applies to arbitrary pairs of ideal primitives – e.g., a fixed-input-length or variable-input length random function or a random permutation. For example, a construction using a permutation which is indifferentiable from a random oracle transforms $\mathsf{psPRP}[\mathcal{S}^{\star\mathsf{rs}}]$-secure permutations into $\mathsf{UCE}[\mathcal{S}^{\star\mathsf{rs}}]$-secure functions. Through such a general treatment, our above result on sponges is a corollary of the indifferentiability analysis of [15].

Second, we show that a weaker version of indifferentiability, which we call *CP-sequential indifferentiability*, suffices. Recall that indifferentiability of a construction M transforming an ideal primitive $\mathbf{I}$ into an ideal primitive $\mathbf{J}$ means that there exists a simulator Sim such that $(\mathsf{M}^{\mathbf{I}}, \mathbf{I})$ and $(\mathbf{J}, \mathsf{Sim}^{\mathbf{J}})$ are indistinguishable. CP-sequential indifferentiability only demands this for distinguishers that make all of their *construction* queries to $\mathsf{M}^{\mathbf{I}}$ / $\mathbf{J}$ *before* they proceed to *primitive* queries to $\mathbf{I}$ / $\mathsf{Sim}^{\mathbf{J}}$. As we will see, this significantly enlarges the set of constructions this result applies to. For example, truncating the permutation output to $r < n$ bits does not achieve indifferentiability, because a simulator on an inverse query Y needs to fix $\pi^{-1}(Y)$ to some X such that, for the given random function $\rho : \{0,1\}^n \to \{0,1\}^r$, $\rho(X)$ is consistent with Y on the first r bits, which is infeasible. Yet, the same construction *is* CP-sequentially indifferentiable, intuitively because there is no way for a distinguisher to catch an inconsistent random X, as this would require an extra query to ρ. CP-sequential indifferentiability is dual to the sequential indifferentiability notion of Mandal, Patarin, and Seurin [33], which considers the opposite order of construction and primitive queries. In fact, the two notions are incomparable, as we explain below.

Finally, we will also show that under suitable restrictions on the construction M, the result extends from reset-secure sources to unpredictable ones. This will allow to lift our result for truncation to unpredictable sources.

Constructing psPRPs. Obviously, a central question is whether the assumption of being a psPRP is, by itself, attainable. Our general theorem answers this question already – existing indifferentiability result for Feistel constructions [29,21,22,23] imply already that the 8-round Feistel construction transforms a function which is $\mathsf{UCE}[\mathcal{S}^{\star\mathsf{rs}}]$-secure into a $\mathsf{psPRP}[\mathcal{S}^{\star\mathsf{rs}}]$-secure permutation.

It is important however to assess whether simpler constructions achieve this result. *Here, we show that the five-round Feistel construction suffices.* Our proof heavily exploits our connection to CP-indifferentiability. Indeed, the six-round lower bound of [21] does not apply for CP-indifferentiabiliy, as it requires the ability to ask construction queries *after* primitive queries, and we show that CP-indifferentiability is achieved at five rounds already. Our result is not merely a simplification of earlier ones, and our simulation strategy is novel. In particular, while we still follow the chain-completion approach of previous works, due to the low number of rounds, we need to introduce new techniques to bound the complexity of the simulator. To our rescue will come the fact that no construction

queries can be made after primitive queries, and hence only a limited number of chain types will need to be completed.

We also note that the we are not aware of any obvious lower bound that shows that more than four rounds are really necessary – four rounds are necessary alone to reach PRP security in the eyes of the source. We leave it as an open problem to show whether four rounds are sufficient. We also note that our result only deals with reset-secure sources, and we leave it as an open problem to find a similar result for unpredictable sources. For reasons we explain in the body, it seems reasonable to conjecture that a heavily unbalanced Feistel network with $\Omega(n)$ rounds achieves this transformation.

Constructing psPRPs, in ideal models. While the main purpose of the psPRP framework is that of removing ideal model assumptions, it is still valuable to assess how psPRPs are built in the first place. To this end, we also show how to *heuristically* instantiate psPRPs from existing cryptographic primitives, and here validation takes us necessarily back to ideal models. Plus, for ideal-model applications that require psPRP security as an intermediate notion (for instance, because we are analyzing two-stage games), these provide instantiations.

We validate two strategies: (1) Using a block cipher, and seed it through the key, and (2) Using a *keyless* permutation, and seeding via the Even-Mansour construction [25]. We prove that the first approach is $\mathsf{psPRP}[\mathcal{S}^{\mathsf{crs}}]$-secure in the ideal-cipher model, and prove the second $\mathsf{psPRP}[\mathcal{S}^{\mathsf{cup}}]$-secure in the random permutation model.[4]

Fixed-key block-cipher based garbling from psPRPs. As a benchmark for psPRPs, we revisit the garbling schemes from [10] based on fixed-key block ciphers, which achieve high degrees of efficiency by eliminating re-keying costs. Their original security analysis was in the ideal-cipher model, and their simplicity is unmatched by schemes with standard-model reductions.

We consider a simple variant of their Ga scheme and prove it secure under the assumption the underlying block cipher, when seeded through its key input, is $\mathsf{psPRP}[\mathcal{S}^{\mathsf{sup}}]$-secure. Our construction is slightly less efficient than the scheme from [10], since a different seed/key is used for every garbling. However, we still gain from the fact that no re-keying is necessary throughout a garbling operation, or the evaluation of a garbled circuit. We also note that our approach also extends to the GaX scheme of [10] with further optimizations.

Extra related work. A few works gave UCE constructions. Brzuska and Mittelbach [18] gave constructions from auxiliary-input point obfuscation (AIPO) and iO. In a recent paper, under the exponential DDH assumption, Zhandry [41] built a primitive (called an AI-PRG) which is equivalent to a UCE for a subset of $\mathcal{S}^{\mathsf{cup}}$ which is sufficient for instantiating point obfuscators. (The observation is not made explicit in [41], but the definitions are equivalent.) None of these results is sufficiently strong to instantiate our Feistel-based construction of psPRPs.

The cryptanalysis community has studied block-cipher security under known keys, albeit with a different focus. For example, Knuden and Rijmen [30] gave

[4] Again, recall that IO-based impossibility for $\mathcal{S}^{\mathsf{cup}}$ and $\mathcal{S}^{\mathsf{crs}}$ do not apply because we are in ideal models.

attacks against Feistel networks and reduced-round versions of AES that find input-output pairs with specific properties (given the key) in time faster than it should be possible if the block cipher were a random permutation. Several such attacks were later given for a number of block ciphers. We are not aware of these attacks however invalidating psPRP security. Andreeva, Bogdanov, and Mennink [2] gave formal models for known-key security in ideal models based on a weak form of indifferentiability, where construction queries are to the construction *under a known random key*. These are however unrelated.

Outline. Section 2 proposes a general framework for public-seed pseudorandom notions, and Section 3 puts this to use to provide general reduction theorems between pairs of such primitives, and defines in particular CP-sequential indifferentiability. UCE constructions from psPRPs are given in Section 4, whereas Section 5 presents our main result on building psPRPs via the Feistel construction. Heuristic constructions are presented in Section 6, and finally we apply psPRPs to the analysis of garbling schemes in Section 7.

Notational preliminaries. Throughout this paper, we denote by $\mathsf{Funcs}(X, Y)$ the set of functions $X \to Y$, and in particular use the shorthand $\mathsf{Funcs}(m, n)$ whenever $X = \{0,1\}^m$ and $Y = \{0,1\}^n$. We also denote by $\mathsf{Perms}(X)$ the set of permutations on the set X, and analogously, $\mathsf{Perms}(n)$ denotes the special case where $X = \{0,1\}^n$. We say that a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *negligible* if for all $c \in \mathbb{N}$, there exists a λ_0 such that $f(\lambda) \leq \lambda^{-c}$ for all $\lambda \geq \lambda_0$.

Our security definitions and proofs will often use games, as formalized by Bellare and Rogaway [12]. Typically, our games will have boolean outputs – that is, either true or false – and we use the shorthand $\mathsf{Pr}[\mathsf{G}]$ to denote the probability that a certain game outputs the value true, or occasionally 1 (when the output is binary, rather than boolean).

2 Public-seed Pseudorandomness

We present a generalization of the UCE notion [8], which we term *public-seed pseudorandomness*. We apply this notion to define psPRPs as a special case, but the general treatment will be useful to capture transformations between UCEs and psPRPs in Section 3 via one single set of theorems.

2.1 Ideal Primitives and their Implementations

We begin by formally defining *ideal primitives* using notation inspired by [27,6].

Ideal primitives. An *ideal primitive* is a pair $\mathbf{I} = (\Sigma, \mathcal{T})$, where $\Sigma = \{\Sigma_\lambda\}_{\lambda \in \mathbb{N}}$ is a family of sets of functions (such that all functions in Σ_λ have the same domain and range), and $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda \in \mathbb{N}}$ is a family of probability distributions, where $\mathcal{T}_\lambda$'s range is a subset of Σ_λ for all $\lambda \in \mathbb{N}$. The ideal primitive $\mathbf{I}$, once the security parameter λ is fixed, should be thought of as an oracle that initially samples a function I as its initial state according to $\mathcal{T}_\lambda$ from Σ_λ. We denote this

sampling as $I \leftarrow\!\!\$\, \mathbf{I}_\lambda$. Then, $\mathbf{I}$ provides access to I via queries, that is, on input $\mathbf{x}$ it returns $I(\mathbf{x})$.[5]

EXAMPLES. We give a few examples of ideal primitives using the above notation. In particular, let $\kappa, m, n : \mathbb{N} \to \mathbb{N}$ be functions.

Example 1. The *random function* $\mathbf{R}_{m,n} = (\Sigma^{\mathbf{R}}, \mathcal{T}^{\mathbf{R}})$ is such that for all $\lambda \in \mathbb{N}$, $\Sigma^{\mathbf{R}}_\lambda = \mathsf{Funcs}(m(\lambda), n(\lambda))$, and $\mathcal{T}^{\mathbf{R}}_\lambda$ is the uniform distribution on $\Sigma^{\mathbf{R}}_\lambda$. We also define $\mathbf{R}_{*,n}$ to be the same for $\mathsf{Funcs}(*, n(\lambda))$, that is, when the domain is extended to arbitrary length input strings.[6]

Example 2. The *random permutation* $\mathbf{P}_n = (\Sigma^{\mathbf{P}}, \mathcal{T}^{\mathbf{P}})$ is such that for all $\lambda \in \mathbb{N}$,

$$\Sigma^{\mathbf{P}}_\lambda = \Big\{ P : \{+,-\} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)} \mid \exists \pi \in \mathsf{Perms}(n(\lambda)) :\ P(+,x) = \pi(x), P(-,x) = \pi^{-1}(x) \Big\} ,$$

and moreover, $\mathcal{T}^{\mathbf{P}}_\lambda$ is the uniform distribution on $\Sigma^{\mathbf{P}}_\lambda$.

Example 3. The *ideal cipher* $\mathbf{IC}_{\kappa,n} = (\Sigma^{\mathbf{IC}}, \mathcal{T}^{\mathbf{IC}})$ is such that

$$\Sigma^{\mathbf{IC}}_\lambda = \Big\{ E : \{0,1\}^{\kappa(\lambda)} \times \{+,-\} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)} \mid \forall k \in \{0,1\}^{\kappa(\lambda)} \exists \pi_k \in \mathsf{Perms}(n(\lambda)) :\ E(k,+,x) = \pi_k(x), E(k,-,x) = \pi_k^{-1}(x) \Big\} ,$$

and $\mathcal{T}^{\mathbf{IC}}_\lambda$ is the uniform distribution on $\Sigma^{\mathbf{IC}}_\lambda$.

EFFICIENCY CONSIDERATIONS. Usually, for an ideal primitive $\mathbf{I} = (\Sigma, \mathcal{T})$, the bit-size of the elements of Σ_λ grows exponentially in λ, and thus one would not implement a primitive $\mathbf{I}$ by sampling I from Σ_λ, but rather using techniques such as lazy sampling. An implementation of a primitive $\mathbf{I}$ is a *stateful* randomized PPT algorithm A such that $A(1^\lambda, \cdot)$ behaves as $I \xleftarrow{\$} \mathbf{I}_\lambda$ for all $\lambda \in \mathbb{N}$. We say that $\mathbf{I}$ is *efficiently implementable* if such an A exists. All the above examples – $\mathbf{R}_{m,n}, \mathbf{R}_{*,n}, \mathbf{P}_n$, and $\mathbf{IC}_{\kappa,n}$ – are efficiently implementable as long as m, n, κ are polynomially bounded functions.

Σ-COMPATIBLE FUNCTION FAMILIES. A *function family* $\mathsf{F} = (\mathsf{Kg}, \mathsf{Eval})$ consists of a *key (or seed) generation algorithm* $\mathsf{F.Kg}$ and an *evaluation algorithm* $\mathsf{F.Eval}$. In particular, $\mathsf{F.Kg}$ is a randomized algorithm that on input the unary representation of the security parameter λ returns a *key* k, and we let $[\mathsf{F.Kg}(1^\lambda)]$ denote the set of all possible outputs of $\mathsf{F.Kg}(1^\lambda)$. Moreover, $\mathsf{F.Eval}$ is a deterministic algorithm that takes three inputs; the security parameter in unary form 1^λ, a key $k \in [\mathsf{F.Kg}(1^\lambda)]$ and a query $\mathbf{x}$ such that $\mathsf{F.Eval}(1^\lambda, k, \cdot)$ implements a function that maps queries $\mathbf{x}$ to $\mathsf{F.Eval}(1^\lambda, k, \mathbf{x})$. We say that F is *efficient* if both Kg and Eval are polynomial-time algorithms.

[5] The reader may wonder whether defining Σ is necessary, but this will allow us to enforce a specific format on valid implementations below.

[6] Note that this requires some care, because Σ_λ is now uncountable, and thus sampling from it requires a precise definition. We will not go into formal details, similar to many other papers, but it is clear that this can easily be done.

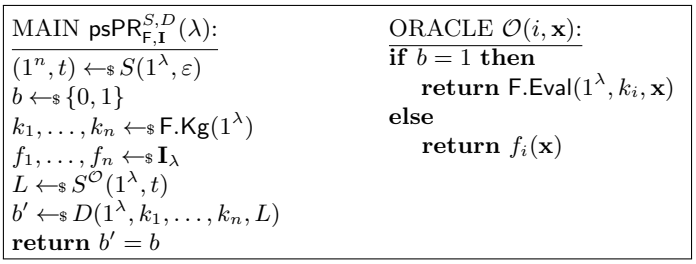

Fig. 2: Game psPR used to define pspr-security for a primitive F that is Σ-compatible with $\mathbf{I}$. Here, S is the source and D is the distinguisher. Recall that the notation $f \leftarrow\!\!\$\ \mathbf{I}_\lambda$ indicates picking a function from Σ_λ using $\mathcal{T}_\lambda$.

Definition 1 (Σ-compatibility). *A function family* F *is Σ-compatible with* $\mathbf{I} = (\Sigma, \mathcal{T})$ *if* $\mathsf{F.Eval}(1^\lambda, k, \cdot) \in \Sigma_\lambda$ *for all* $\lambda \in \mathbb{N}$ *and* $k \in [\mathsf{F.Kg}(1^\lambda)]$.

2.2 Public-seed Pseudorandomness, psPRPs, and Sources

We now define a general notion of public-seed pseudorandom implementations of ideal primitives.

The general definition. Let $\mathsf{F} = (\mathsf{Kg}, \mathsf{Eval})$ be a function family that is Σ-compatible with an ideal primitive $\mathbf{I} = (\Sigma, \mathcal{T})$. Let S be an adversary called the *source* and D an adversary called the *distinguisher*. We associate to them, F and $\mathbf{I}$, the game $\mathsf{psPR}^{S,D}_{\mathsf{F},\mathbf{I}}(\lambda)$ depicted in Fig. 2. The source initially chooses the number of keys n. Then, in the second stage, it is given access to an oracle $\mathcal{O}$ and we require any query $(i, \mathbf{x})$ made to this oracle be valid, that is, $\mathbf{x}$ is a valid query for any $f_i \in \Sigma_\lambda$ and $i \in [n]$, for n output by the first stage of the source. When the challenge bit $b = 1$ ("real") the oracle responds via $\mathsf{F.Eval}$ under the key k_i ($\mathsf{F.Eval}(1^\lambda, k_i, \cdot)$) that is chosen by the game and *not* given to the source. When $b = 0$ ("ideal") it responds via f_i where $f_i \leftarrow\!\!\$\ \mathbf{I}_\lambda$. After its interaction with the oracle $\mathcal{O}$, the source S communicates the *leakage* $L \in \{0,1\}^*$ to D. The distinguisher is given access to the keys $k_1, \ldots, k_n$ and must now guess $b' \in \{0,1\}$ for b. The game returns true iff $b' = b$ and we describe the pspr-advantage of (S, D) for $\lambda \in \mathbb{N}$ as

$$\mathsf{Adv}^{\mathsf{pspr}[\mathbf{I}]}_{\mathsf{F},S,D}(\lambda) = 2\Pr\left[\mathsf{psPR}^{S,D}_{\mathsf{F},\mathbf{I}}(\lambda)\right] - 1\ . \tag{1}$$

In the following, we are going to use the shorthands $\mathsf{UCE}[m,n]$ for $\mathsf{pspr}[\mathbf{R}_{m,n}]$, $\mathsf{UCE}[n]$ for $\mathsf{pspr}[\mathbf{R}_{*,n}]$, and $\mathsf{psPRP}[n]$ for $\mathsf{pspr}[\mathbf{P}_n]$.

Note that our security game captures the multi-key version of the security notions, also considered in past works on UCE, as it is not known to be implied by the single-key version, which is recovered by having the source initially output $n = 1$.

MAIN $\mathsf{Pred}^P_{\mathbf{I},S}(\lambda)$:	MAIN $\mathsf{Reset}^R_{\mathbf{I},S}(\lambda)$:
$\mathsf{done} \leftarrow \mathsf{false}; Q \leftarrow \emptyset;\ (1^n, t) \leftarrow\!\!\$\ S(1^\lambda, \varepsilon)$	$\mathsf{done} \leftarrow \mathsf{false};\ (1^n, t) \leftarrow\!\!\$\ S(1^\lambda, \varepsilon)$
$f_1, \ldots, f_n \leftarrow\!\!\$\ \mathbf{I}_\lambda$	$f_1^0, f_1^1, \ldots, f_n^0, f_n^1 \leftarrow\!\!\$\ \mathbf{I}_\lambda$
$L \leftarrow\!\!\$\ S^{\mathcal{O}}(1^\lambda, t); \mathsf{done} \leftarrow \mathsf{true}$	$L \leftarrow\!\!\$\ S^{\mathcal{O}}(1^\lambda, t); \mathsf{done} \leftarrow \mathsf{true}$
$Q' \leftarrow\!\!\$\ P^{\mathcal{O}}(1^\lambda, 1^n, L)$	$b \leftarrow\!\!\$\ \{0,1\}; b' \leftarrow\!\!\$\ R^{\mathcal{O}}(1^\lambda, 1^n, L)$
return $(Q \cap Q' \neq \emptyset)$	**return** $b' = b$
ORACLE $\mathcal{O}(i, \mathbf{x})$:	ORACLE $\mathcal{O}(i, \mathbf{x})$:
if $\neg\mathsf{done}$ **then** $Q \leftarrow Q \cup \{\mathbf{x}\}$	**if** $\neg\mathsf{done}$ **then return** $f_i^0(\mathbf{x})$
return $f_i(\mathbf{x})$	**else return** $f_i^b(\mathbf{x})$

Fig. 3: Games Pred and Reset are used to define the unpredictability and reset-security of the source S respectively against the ideal primitive $\mathbf{I}$. Here, S is the source, P is the predictor and R is the reset adversary.

Restricting sources. One would want to define F as secure if $\mathsf{Adv}^{\mathsf{pspr}[\mathbf{I}]}_{\mathsf{F},S,D}(\lambda)$ is negligible in λ for all polynomial time sources S and distinguishers D. However, as shown already in the special case of UCEs [8], this is impossible, as one can always easily construct (at least for non-trivial $\mathbf{I}$'s) a simple source S which leaks the evaluation of $\mathcal{O}$ on a given point, and D can check consistency given k.

Therefore to obtain meaningful and non-empty security definitions we restrict the considered sources to some class $\mathcal{S}$, without restricting the distinguisher class. Consequently, we denote by $\mathsf{psPR}[\mathbf{I}, \mathcal{S}]$ the security notion that asks $\mathsf{Adv}^{\mathsf{pspr}[\mathbf{I}]}_{\mathsf{F},S,D}(\lambda)$ to be negligible for all polynomial time distinguishers D and all sources $S \in \mathcal{S}$. Following [8], we also use $\mathsf{psPR}[\mathbf{I}, \mathcal{S}]$ to denote the set of F's which are $\mathsf{psPR}[\mathbf{I}, \mathcal{S}]$-secure. Note that obviously, if $\mathcal{S}_1 \subseteq \mathcal{S}_2$, then $\mathsf{psPR}[\mathbf{I}, \mathcal{S}_2] \subseteq \mathsf{psPR}[\mathbf{I}, \mathcal{S}_1]$ where $\mathcal{S}_1$ and $\mathcal{S}_2$ are source classes for the ideal primitive $\mathbf{I}$. We will use the shorthands $\mathsf{psPRP}[n, \mathcal{S}]$ for $\mathsf{psPR}[\mathbf{P}_n, \mathcal{S}]$ and $\mathsf{UCE}[m, n, \mathcal{S}]$ for $\mathsf{psPR}[\mathbf{R}_{m,n}, \mathcal{S}]$, where $m = *$ if the domain is unbounded.

Below, we discuss two important classes of restrictions, which are fundamental for the remainder of this paper – unpredictable and reset-secure sources.

Unpredictable sources. Let S be a source. Consider the game $\mathsf{Pred}^P_{\mathbf{I},S}(\lambda)$ of Fig. 3 associated to S and an adversary P called the predictor. Given the leakage, the latter outputs a set Q'. It wins if this set contains any $\mathcal{O}$-query of the source. For $\lambda \in \mathbb{N}$ we let

$$\mathsf{Adv}^{\mathsf{pred}[\mathbf{I}]}_{S,P}(\lambda) = \Pr\left[\mathsf{Pred}^P_{\mathbf{I},S}(\lambda)\right] . \tag{2}$$

We say that P is a *computational predictor* if it is polynomial time, and it is a *statistical* predictor if there exists polynomials q, q' such that for all $\lambda \in \mathbb{N}$, predictor P makes at most $q(\lambda)$ oracle queries and outputs a set Q' of size at most $q'(\lambda)$ in game $\mathsf{Pred}^P_{\mathbf{I},S}(\lambda)$. We stress that in this case the predictor need not be polynomial time, even though it makes a polynomial number of queries. We say S is *computationally unpredictable* if $\mathsf{Adv}^{\mathsf{pred}[\mathbf{I}]}_{S,P}(\lambda)$ is negligible for all computational predictors P. We say S is *statistically unpredictable* if $\mathsf{Adv}^{\mathsf{pred}[\mathbf{I}]}_{S,P}(\lambda)$ is

negligible for all statistical predictors P. We let $\mathcal{S}^{\mathsf{cup}}$ be the class of all polynomial time, computationally unpredictable sources and $\mathcal{S}^{\mathsf{sup}} \subseteq \mathcal{S}^{\mathsf{cup}}$ be the class of all polynomial time statistically unpredictable sources.[7]

RESET-SECURE SOURCES. Let S be a source. Consider the game $\mathsf{Reset}^R_{\mathbf{I},S}(\lambda)$ of Fig. 3 associated to S and an adversary R called the reset adversary. The latter wins if given the leakage L it can distinguish between f^0 used by the source S and an independent f^1 where $f^0, f^1 \leftarrow\!\!\$\ \mathbf{I}_\lambda$. For $\lambda \in \mathbb{N}$ we let

$$\mathsf{Adv}^{\mathsf{reset}[\mathbf{I}]}_{S,R}(\lambda) = 2\Pr\left[\mathsf{Reset}^R_{\mathbf{I},S}(\lambda)\right] - 1\ . \tag{3}$$

We say that R is a *computational reset adversary* if it is polynomial time, and it is a statistical reset adversary if there exists a polynomial q such that for all $\lambda \in \mathbb{N}$, reset adversary R makes at most $q(\lambda)$ oracle queries in game $\mathsf{Reset}^R_{\mathbf{I},S}(\lambda)$. We stress that in this case the reset adversary need not be polynomial time. We say S is *computationally reset-secure* if $\mathsf{Adv}^{\mathsf{reset}[\mathbf{I}]}_{S,R}(\lambda)$ is negligible for all computational reset adversaries R. We say S is *statistically reset-secure* if $\mathsf{Adv}^{\mathsf{reset}[\mathbf{I}]}_{S,R}(\lambda)$ is negligible for all statistical reset adversaries R. We let $\mathcal{S}^{\mathsf{crs}}$ be the class of all polynomial time, computationally reset-secure sources and $\mathcal{S}^{\mathsf{srs}} \subseteq \mathcal{S}^{\mathsf{crs}}$ the class of all polynomial time statistically reset-secure sources.

RELATIONSHIPS. For the case of psPRPs, we mention the following fact, which is somewhat less obvious than in the UCE case, and in particular only holds if the permutation's domain grows with the security parameter.

Proposition 1. *For all $n \in \omega(\log \lambda)$, we have $\mathsf{psPRP}[n, \mathcal{S}^{\star\mathsf{rs}}] \subseteq \mathsf{psPRP}[n, \mathcal{S}^{\star\mathsf{up}}]$ where $\star \in \{\mathsf{c}, \mathsf{s}\}$.*

Proof (Sketch). In the reset secure game, consider the event that R queries its oracle $\mathcal{O}$ on input (i, σ, x) which was queried by S already as an $\mathcal{O}(i, \sigma, x)$ query, or it was the answer to a query $\mathcal{O}(i, \overline{\sigma}, y)$. Here (like elsewhere in the paper), we use the notational convention $\overline{+} = -$ and $\overline{-} = +$. The key point here is proving that as long as this bad event does not happen, the $b = 0$ and $b = 1$ case are hard to distinguish. A difference with the UCE case is that due to the permutation property, they will not be perfectly indistinguishable, but a fairly standard (yet somewhat tedious) birthday argument suffices to show that indistinguishability still holds as long as the overall number of $\mathcal{O}$ queries (of S and R) is below $2^{n(\lambda)/2}$, which is super-polynomial for $n(\lambda) = \omega(\log \lambda)$. □

3 Reductions and Indifferentiability

We present general theorems that we will use to obtain reductions between psPRPs and UCEs. Our general notation for public-seed pseudorandom primitives allows us to capture the reductions through two general theorems.

[7] We note that computational unpredictability is only meaningful for sufficiently restricted classes of sources or in ideal models, as otherwise security against $\mathcal{S}^{\mathsf{cup}}$ is not achievable assuming IO, using essentially the same attack as [17].

MAIN $\mathsf{CP}[\mathbf{I} \to \mathbf{J}]^A_{\mathsf{M},\mathsf{Sim}}(\lambda)$:	ORACLE $\mathsf{Func}(\mathbf{x})$:	ORACLE $\mathsf{Prim}(\mathbf{u})$:
$b \leftarrow\!\!\$\ \{0,1\}$; $f \leftarrow\!\!\$\ \mathbf{I}_\lambda$; $g \leftarrow\!\!\$\ \mathbf{J}_\lambda$	**if** $b = 1$ **then**	**if** $b = 1$ **then**
$\mathsf{st} \leftarrow\!\!\$\ A_1^{\mathsf{Func}}(1^\lambda)$	**return** $\mathsf{M}^f(\mathbf{x})$	**return** $f(\mathbf{u})$
$b' \leftarrow\!\!\$\ A_2^{\mathsf{Prim}}(1^\lambda, \mathsf{st})$	**else**	**else**
return $b' = b$	**return** $g(\mathbf{x})$	**return** $\mathsf{Sim}^g(\mathbf{u})$

Fig. 4: Game CP used to define cpi-security for a construction M implementing the primitive **J** using primitive **I**. Here, Sim is the simulator and $A = (A_1, A_2)$ is the two-stage distinguisher.

CP-SEQUENTIAL INDIFFERENTIABILITY. Indifferentiability was introduced in [35] by Maurer, Renner, and Holenstein to formalize reductions between ideal primitives. Following their general treatment, it captures the fact that a (key-less) construction M using primitive **I** (which can be queried by the adversary directly) is *as good* as another primitive **J** by requiring the existence of a simulator that can simulate **I** consistently by querying **J**.

Central to this paper is a weakening of indifferentiability that we refer to as *CP-sequential indifferentiability*, where the distinguisher A makes all of its *construction* queries to $\mathsf{M}^{\mathbf{I}}$ (or **J**) *before* moving to making *primitive queries* to **I** (or $\mathsf{Sim}^{\mathbf{J}}$, where Sim is the simulator). Note that this remains a non-trivial security goal, since Sim does not learn the construction queries made by A, but needs to simulate correctly nonetheless. However, the hope is that because A has committed to its queries before starting its interaction with Sim, the simulation task will be significantly easier. (We will see that this is indeed the case.)

More concretely, the notion is concerned with constructions which implement **J** from **I**, and need to at least satisfy the following syntactical requirement.

Definition 2 ($(\mathbf{I} \to \mathbf{J})$-compatibility). *Let* $\mathbf{I} = (\mathbf{I}.\Sigma, \mathbf{I}.\mathcal{T})$ *and* $\mathbf{J} = (\mathbf{J}.\Sigma, \mathbf{J}.\mathcal{T})$ *be ideal primitives. A construction* M *is called* $(\mathbf{I} \to \mathbf{J})$*-compatible if for every* $\lambda \in \mathbb{N}$*, and every* $f \in \mathbf{I}.\Sigma_\lambda$*, the construction* M *implements a function* $x \mapsto \mathsf{M}^f(1^\lambda, x)$ *which is in* $\mathbf{J}.\Sigma_\lambda$*.*

The game CP is described in Fig. 4. For ideal primitives $\mathbf{I}, \mathbf{J}$, a two-stage adversary $A = (A_1, A_2)$, an $(\mathbf{I} \to \mathbf{J})$-compatible construction M, and simulator Sim, as well as security parameter $\lambda \in \mathbb{N}$, we define

$$\mathsf{Adv}^{\mathsf{cpi}[\mathbf{I}\to\mathbf{J}]}_{\mathsf{M},\mathsf{Sim},A}(\lambda) = 2 \cdot \Pr\left[\mathsf{CP}[\mathbf{I} \to \mathbf{J}]^A_{\mathsf{M},\mathsf{Sim}}(\lambda)\right] - 1\ . \tag{4}$$

We remark that the CP-sequential indifferentiability notion is the exact dual of sequential indifferentiability as introduced by Mandal, Patarin, and Seurin [33], which postpones construction queries *to the end*. As we will show below in Section 4.2, there are CP-indifferentiable constructions which are not sequentially indifferentiable in the sense of [33].

REDUCTIONS. We show that CP-sequential indifferentiability yields a reduction between public-seed pseudorandomness notions. A special case was shown in [9]

by Bellare, Hoang, and Keelvedhi for domain extension of UCEs. Our result goes beyond in that: (1) It is more general, as it deals with arbitrary ideal primitives, (2) It only relies on CP-sequential indifferentiability, as opposed to full indifferentiability, and (3) The reduction of [9] only considered reset-secure sources, whereas we show that under certain conditions on the construction, the reduction also applies to unpredictable sources. Nonetheless, our proofs follow the same approach of [9], and the main contribution is conceptual.

We let $\mathsf{F} = (\mathsf{F.Kg}, \mathsf{F.Eval})$ be a function family which is Σ-compatible with an ideal primitive $\mathbf{I}$. Further, let M be an $(\mathbf{I} \to \mathbf{J})$-compatible construction. Then, overloading notation, we define the new function family $\mathsf{M}[\mathsf{F}] = (\mathsf{M.Kg}, \mathsf{M.Eval})$, where $\mathsf{M.Kg} = \mathsf{F.Kg}$, and for every $k \in [\mathsf{M.Kg}(1^\lambda)]$, we let

$$\mathsf{M.Eval}(1^\lambda, k, x) = \mathsf{M}^O(1^\lambda, x) \ , \tag{5}$$

where $O(z) = \mathsf{F.Eval}(1^\lambda, k, z)$.

Reset-secure sources. The following is our general reduction theorem for the case of reset-secure sources. Its proof follows similar lines as the one in [9] and we refer the reader to the full version for details.

Theorem 1 (Composition theorem, reset-secure case). *Let* M, F, $\mathbf{I}$, *and* $\mathbf{J}$ *be as above. Fix any simulator* Sim. *Then, for every source-distinguisher pair* (S, D), *where* S *requests at most* $N(\lambda)$ *keys, there exists a source-distinguisher pair* $(\overline{S}, \overline{D})$, *and a further distinguisher* A, *such that*

$$\mathsf{Adv}^{\mathsf{pspr}[\mathbf{J}]}_{\mathsf{M}[\mathsf{F}],S,D}(\lambda) \leq \mathsf{Adv}^{\mathsf{pspr}[\mathbf{I}]}_{\mathsf{F},\overline{S},\overline{D}}(\lambda) + N(\lambda) \cdot \mathsf{Adv}^{\mathsf{cpi}[\mathbf{I}\to\mathbf{J}]}_{\mathsf{M},\mathsf{Sim},A}(\lambda) \ . \tag{6}$$

Here, in particular: The complexities of D *and* $\overline{D}$ *are the same. Moreover, if* S, D, *and* M *are polynomial time, and* $\mathbf{I}$, $\mathbf{J}$ *are efficiently implementable, then* A, $\overline{S}$ *and* $\overline{D}$ *are also polynomial-time.*

Moreover, for every reset adversary R, *there exists a reset adversary* R' *and a distinguisher* B *such that*

$$\mathsf{Adv}^{\mathsf{reset}[\mathbf{I}]}_{\overline{S},R}(\lambda) \leq \mathsf{Adv}^{\mathsf{reset}[\mathbf{J}]}_{S,R'}(\lambda) + 3N(\lambda) \cdot \mathsf{Adv}^{\mathsf{cpi}[\mathbf{I}\to\mathbf{J}]}_{\mathsf{M},\mathsf{Sim},B}(\lambda) \ , \tag{7}$$

where R' *makes a polynomial number of query / runs in polynomial time if* R *and* Sim *make a polynomial number of queries / run in polynomial time, and* $\mathbf{I}, \mathbf{J}$ *are efficiently implementable.* ■

Query extractable constructions. Next, we show that under strong conditions on the construction M, Theorem 1 extends to the case of unpredictability.

In particular, we consider constructions which we term *query extractable*. Roughly, what such constructions guarantee is that every query made by M to an underlying ideal primitive $\mathbf{I}$ can be assigned to a (small) set of possible inputs to M that would result in this query during evaluation. Possibly, this set of inputs may be found by making some additional queries to $\mathbf{I}$. We define this formally through the game $\mathsf{EXT}^{S,P}_{\mathsf{M},\mathbf{I},\mathsf{Ext}}(\lambda)$ in Fig. 5. It involves a *source* S and a *predictor*

GAME $\mathsf{EXT}^{S,P}_{\mathsf{M},\mathbf{I},\mathsf{Ext}}(\lambda)$:	ORACLE $\mathcal{O}(i,x)$:
$\mathsf{done} \leftarrow \mathsf{false}$	**if** $\neg\mathsf{done}$ **then** $Q_{\mathbf{I}} \xleftarrow{\cup} \{x\}$
$Q_{\mathbf{I}}, Q_{\mathsf{M}} \leftarrow \emptyset$	**return** $f_i(x)$
$(1^n, \mathsf{st}) \leftarrow\!\!\$\ S(1^\lambda, \varepsilon)$	
$f_1, \ldots, f_n \leftarrow\!\!\$\ \mathbf{I}_\lambda$	ORACLE $\mathcal{O}_{\mathsf{M}}(i,x)$:
$L \leftarrow\!\!\$\ S^{\mathcal{O}_{\mathsf{M}}}(1^\lambda, 1^n, \mathsf{st})$	**if** $\neg\mathsf{done}$ **then** $Q_{\mathsf{M}} \xleftarrow{\cup} \{x\}$
$\mathsf{done} \leftarrow \mathsf{true}$	$y \leftarrow \mathsf{M}^{\mathcal{O}(i,\cdot)}(x)$
$Q \leftarrow\!\!\$\ P^{\mathcal{O}}(1^\lambda, 1^n, L)$; $Q^* \leftarrow \mathsf{Ext}^{\mathcal{O}}(Q)$	**return** y
return $((Q \cap Q_{\mathbf{I}} \neq \emptyset) \wedge (Q^* \cap Q_{\mathsf{M}} = \emptyset))$	

Fig. 5: Game $\mathsf{EXT}^{S,P}_{\mathsf{M},\mathbf{I},\mathsf{Ext}}(\lambda)$ in the definition of query extractability.

P, as well as an extractor Ext. Here, S selects an integer n, which results in n instances $f_1, \ldots, f_n$ of $\mathbf{I}$ being spawned, and then makes queries to n instances of M^{f_i}, gives some leakage to the predictor P, and the predictor makes further query to the $\mathbf{I}$-instances, until it outputs a set Q. Then, we run the extractor Ext on Q, and the extractor can also make additional queries to the $\mathbf{I}$-instances, and outputs an additional set Q^*. We are interested in the event that Q contains one of queries made to the f_i's by M in the first stage of the game, yet Q^* does not contain any of S's queries to M^{f_i} for some i. In particular, we are interested in

$$\mathsf{Adv}^{\mathsf{ext}[\mathbf{I}]}_{\mathsf{M},S,P,\mathsf{Ext}}(\lambda) = \mathsf{Pr}\left[\mathsf{EXT}^{S,P}_{\mathsf{M},\mathbf{I},\mathsf{Ext}}(\lambda)\right] .$$

We say that M is *query extractable with respect to* $\mathbf{I}$ if there exists a polynomial time Ext such that $\mathsf{Adv}^{\mathsf{ext}[\mathbf{I}]}_{\mathsf{M},S,P,\mathsf{Ext}}(\lambda)$ is negligible for all PPT P and S. We say it is *perfectly* query extractable if the advantage is 0, rather than simply negligible.

The next theorem provides an alternative to Theorem 1 for the case of unpredictable sources whenever M guarantees query extractability.

Theorem 2 (Composition theorem, unpredictable case). *Let* M, F, $\mathbf{I}$, *and* $\mathbf{J}$ *be as before. Fix any simulator* Sim. *Then, for every source-distinguisher pair* (S, D), *where* S *requests at most* $N(\lambda)$ *keys, there exists a source-distinguisher pair* $(\overline{S}, \overline{D})$, *and a further distinguisher* A, *such that*

$$\mathsf{Adv}^{\mathsf{pspr}[\mathbf{J}]}_{\mathsf{M}[\mathsf{F}],S,D}(\lambda) \leq \mathsf{Adv}^{\mathsf{pspr}[\mathbf{I}]}_{\mathsf{F},\overline{S},\overline{D}}(\lambda) + N(\lambda) \cdot \mathsf{Adv}^{\mathsf{cpi}[\mathbf{I}\to\mathbf{J}]}_{\mathsf{M},\mathsf{Sim},A}(\lambda) . \tag{8}$$

Here, in particular: The complexities of D *and* $\overline{D}$ *are the same. Moreover, if* S, D, *and* M *are polynomial time, and* $\mathbf{I}$, $\mathbf{J}$ *are efficiently implementable, then* A, $\overline{S}$ *and* $\overline{D}$ *are also polynomial-time.*

Moreover, for every predictor P *and extractor* Ext, *there exists a predictor adversary* P' *and a distinguisher* B *such that*

$$\mathsf{Adv}^{\mathsf{pred}[\mathbf{I}]}_{\overline{S},P}(\lambda) \leq \mathsf{Adv}^{\mathsf{pred}[\mathbf{J}]}_{S,P'}(\lambda) + \mathsf{Adv}^{\mathsf{ext}[\mathbf{I}]}_{\mathsf{M},S,P,\mathsf{Ext}}(\lambda) + N(\lambda) \cdot \mathsf{Adv}^{\mathsf{cpi}[\mathbf{I}\to\mathbf{J}]}_{\mathsf{M},\mathsf{Sim},B}(\lambda) , \tag{9}$$

where P' *makes a polynomial number of query / runs in polynomial time if* P, Sim *and* Ext *make a polynomial number of queries / run in polynomial time, and* $\mathbf{I}, \mathbf{J}$ *are efficiently implementable.* ■

4 From psPRPs to UCEs

We consider the problem of building UCEs from psPRPs. On the one hand, we want to show that all applications of UCEs can be recovered modularly by instantiating the underlying UCE with a psPRP-based construction. Second, we want to show that practical permutation-based designs can be instantiated by assuming the underlying permutation (when equipped with a seed) is a psPRP.

4.1 Reset-secure Sources and Sponges

The case of reset-secure sources follows by a simple application of Theorem 1: A number of constructions from permutations have been proved indifferentiable from a random oracle, and all of these yield a construction of a UCE for $\mathcal{S}^{\star\mathsf{rs}}$ when the underlying permutation is a psPRP for $\mathcal{S}^{\star\mathsf{rs}}$, where $\star \in \{\mathsf{c},\mathsf{s}\}$.[8]

SPONGES. A particular instantiation worth mentioning is the sponge construction by Bertoni et al. [15], which underlies KECCAK/SHA-3. In particular, let $\mathsf{Sponge}_{n,r}$ be the $(\mathbf{P}_n \to \mathbf{R}_{*,r})$-compatible construction which operates as follows, on input 1^λ, $M \in \{0,1\}^*$, and given oracle access to a permutation $\rho : \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$. The message M is split into r-bit blocks $M[1],\ldots,M[\ell]$, and the computation keeps a state $S_i \,\|\, T_i$, where $S_i \in \{0,1\}^r$ and $T_i \in \{0,1\}^{n-r}$. Then, $\mathsf{Sponge}^{\rho}_{n,r}(1^\lambda, M) = S_\ell[1..r]$, where

$$S_0 \,\|\, T_0 \leftarrow 0^n \ , \quad S_i \,\|\, T_i \leftarrow \rho((S_{i-1} \oplus M[i]) \,\|\, T_{i-1}) \text{ for } i = 1,\ldots,\ell \ .$$

Then, the following theorem follows directly from Theorem 1 and the indifferentiability analysis of [15]. (We state here only the asymptotic version, but concrete parameters can be obtained from these theorems.)

Theorem 3 (UCE-security for Sponges). *For* $\star \in \{\mathsf{c},\mathsf{s}\}$ *and* $n(\lambda)$ *polynomially bounded in* λ*, if* $\mathsf{F} \in \mathsf{psPRP}[n, \mathcal{S}^{\star\mathsf{rs}}]$*, then* $\mathsf{Sponge}_{n,r}[\mathsf{F}] \in \mathsf{UCE}[*, r, \mathcal{S}^{\star\mathsf{rs}}]$ *whenever* $n(\lambda) - r(\lambda) = \omega(\log \lambda)$. ■

HEURISTIC INSTANTIATION. We wish to say this validates SHA-3 as being a good UCE. One caveat of Theorem 3 is that the actual sponge construction (as used in SHA-3) uses a seedless permutation π. We propose the following assumption on such a permutation π that – if true – implies a simple way to modify an actual Sponge construction to be a secure UCE using Theorem 3. In particular, we suggest using the Even-Mansour [25] paradigm to add a seed to π. Given a family of permutations $\mathit{\Pi} = \{\pi_\lambda\}_{\lambda\in\mathbb{N}}$, where $\pi_\lambda \in \mathsf{Perms}(n(\lambda))$, define then $\mathsf{EM}[\mathit{\Pi}] = (\mathsf{EM.Kg}, \mathsf{EM.Eval})$ where $\mathsf{EM.Kg}$ outputs a random $n(\lambda)$-bit string s on input 1^λ, and

$$\mathsf{EM.Eval}(1^\lambda, s, (+, x)) = s \oplus \pi_\lambda(x \oplus s) \ , \ \mathsf{EM.Eval}(1^\lambda, s, (-, y)) = s \oplus \pi_\lambda^{-1}(y \oplus s)$$

[8] One caveat is that some of these constructions use a few independent random permutations, whereas Theorem 1 assumes only one permutation is used. We point out in passing that Theorem 1 can easily be adapted to this case.

for all $s, x \in \{0,1\}^{n(\lambda)}$. Now, if Π is such that $\mathsf{EM}[\Pi]$ is $\mathsf{psPRP}[n, \mathcal{S}^{\mathsf{srs}}]$-secure, then $\mathsf{Sponge}[\mathsf{EM}[\Pi]]$ is $\mathsf{UCE}[*, r, \mathcal{S}^{\mathsf{srs}}]$-secure by Theorem 3. We discuss the conjecture that EM is $\mathsf{psPRP}[n, \mathcal{S}^{\mathsf{srs}}]$-secure further below in Section 6.

The attractive feature of $\mathsf{Sponge}[\mathsf{EM}[\Pi]]$ is that it can be implemented in a (near) black-box way from $\mathsf{Sponge}[\Pi]$, that is, the original sponge construction run with fixed oracle Π, by setting (1) The initial state $S_0 \,\|\, T_0$ to the seed s (rather than $0^{n(\lambda)}$), and (2) xoring the first r bits $s[1 \dots r]$ of the seed s to the output. The other additions of the seed s to the inner states are unnecessary, as they cancel out. (A similar observation was made by Chang et al [19] in the context of keying sponges to obtain PRFs.)

4.2 Unpredictable Sources

Many UCE applications only require (statistical) unpredictability. In this section, we see that for this weaker target a significantly simpler construction can be used. In particular, we will first build a $\mathsf{UCE}[n, r, \mathcal{S}^{\star\mathsf{up}}]$-secure *compression function* from a $\mathsf{psPRP}[n, \mathcal{S}^{\star\mathsf{up}}]$-secure permutation, where $n(\lambda) - r(\lambda) = \omega(\log \lambda)$ and $\star \in \{\mathsf{c}, \mathsf{s}\}$. Combined with existing domain extension techniques [9], this can be enhanced to a variable-input-length UCE for the same class of sources.

THE CHOP CONSTRUCTION. Let $r, n : \mathbb{N} \to \mathbb{N}$ be polynomially bounded functions of the security parameter λ, where $r(\lambda) \leq n(\lambda)$ for all $\lambda \in \mathbb{N}$. We consider the following construction $\mathsf{Chop}[n, r]$ which is $(\mathbf{P}_n \to \mathbf{R}_{n,r})$-compatible. On input 1^λ, it expects a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ for $n = n(\lambda)$, and given additionally $x \in \{0,1\}^{n(\lambda)}$, it returns

$$\mathsf{Chop}[n, r]^\pi(1^\lambda, x) = \pi(x)[1 \dots r(\lambda)] \,, \tag{10}$$

that is, the first $r = r(\lambda)$ bits of $\pi(x)$. It is not hard to see that the construction is (perfectly) query extractable using the extractor Ext which given oracle access to $\mathcal{O}$ and a set Q of queries of the form $(+, x)$ and $(-, y)$, returns a set consisting of all x such that $(+, x) \in Q$, and moreover adds x' to the set obtained by querying $\mathcal{O}(i, -, y)$ for every $i \in [n]$ and $(-, y) \in Q$.

CP-SEQUENTIAL INDIFFERENTIABILITY. The following theorem establishes CP-sequential indifferentiability of the Chop construction. We refer the reader to the full version for the proof but give some intuition about it after the theorem.

Theorem 4 (CP-indifferentiability of Chop). *Let $r, n : \mathbb{N} \to \mathbb{N}$ be such that $r(\lambda) \leq n(\lambda)$ for all $\lambda \in \mathbb{N}$. Let $\mathbf{P} = \mathbf{P}_n$ and $\mathbf{R} = \mathbf{R}_{n,r}$ be the random permutation and random function, respectively. Then, there exists a simulator Sim such that for all distinguishers A making at most q construction and p primitive queries,*

$$\mathsf{Adv}^{\mathsf{cpi}[\mathbf{P} \to \mathbf{R}]}_{\mathsf{Chop}[n,r], \mathsf{Sim}, A}(\lambda) \leq \frac{(q+p)^2}{2^n} + \frac{p \cdot q}{2^{n-r}} \,. \tag{11}$$

Here, Sim makes at most one oracle query upon each invocation, and otherwise runs in time polynomial in the number of queries answered. ■

The dependence on r is necessary, as otherwise the construction becomes invertible and cannot be CP-sequentially indifferentiable. Also, note that we cannot expect full indifferentiability to hold for the Chop construction, and in fact, not even sequential indifferentiability in the sense of [33]. Indeed, a distinguisher A can simply first query $\mathsf{Prim}(\text{-}, y)$, obtaining x, and then query $\mathsf{Func}(x)$, that yields y'. Then, A just checks that the first r bits of y equals y', and if so outputs 1, and otherwise outputs 0. Note that in the real world, A always outputs 1, by the definition of Chop. However, in the ideal world, an arbitrary simulator Sim needs, on input y, to return an x for which the random oracle (to which it access) returns the first r bits of y. This is however infeasible if $n - r = \omega(\log \lambda)$, unless the simulator can make roughly 2^r queries.

The proof in full version shows this problem vanishes for CP-sequential indifferentiability. Indeed, our simulator will respond to queries $\mathsf{Sim}(\text{-}, y)$ with a random (and inconsistent) x. The key point is that due to the random choice, it is unlikely that the distinguisher has already issued a prior query $\mathsf{Func}(x)$. Moreover, it is also unlikely (in the real world) that the distinguisher, after a query $\mathsf{Func}(x)$, makes an inverse query on $\pi(x)$. The combination of these two facts will be enough to imply the statement.

UCE SECURITY. We can now combine Theorem 4 with the fact that the Chop construction is (perfectly) query extractable, and use Theorem 2:

Corollary 1. *For all n, r such that $n(\lambda) - r(\lambda) = \omega(\log \lambda)$, if F is $\mathsf{psPRP}[n, \mathcal{S}^{\star\mathsf{up}}]$-secure, then $\mathsf{Chop}[\mathsf{F}]$ is $\mathsf{UCE}[n, r, \mathcal{S}^{\star\mathsf{up}}]$-secure, where $\star \in \{\mathsf{c}, \mathsf{s}\}$.*

The construction of [9] can be used to obtain variable-input-length UCE: It first hashes the arbitrary-long input down to an $n(\lambda)$-bit long input using an almost-universal hash function, and then applies $\mathsf{Chop}[\mathsf{F}]$ to the resulting value.

5 Building psPRPs from UCEs

This section presents our main result on building psPRPs from UCEs, namely that the five-round Feistel construction, when its round functions are instantiated from a $\mathsf{UCE}[\mathcal{S}^{\star\mathsf{rs}}]$-secure function family (for $\star \in \{\mathsf{c}, \mathsf{s}\}$), yields a $\mathsf{psPRP}[\mathcal{S}^{\star\mathsf{rs}}]$-secure permutation family.

CP-INDIFFERENTIABILITY OF FEISTEL. Let $n : \mathbb{N} \to \mathbb{N}$ be a (polynomially bounded) function. We define the following construction Ψ_5, which, for security parameter λ, implements an invertible permutation on $2n(\lambda)$-bit strings, and makes calls to an oracle $f : [5] \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$. In particular, on input 1^λ and $X = X_0 \,\|\, X_1$, where $X_0, X_1 \in \{0,1\}^{n(\lambda)}$, running $\Psi_5^f(1^\lambda, (+, X))$ outputs $X_5 \,\|\, X_6$, where

$$X_{i+1} \leftarrow X_{i-1} \oplus f(i, X_i) \text{ for all } i = 1, \ldots, 5\,. \tag{12}$$

Symmetrically, upon an inverse query, $\Psi_5^f(1^\lambda, (\text{-}, Y = X_5 \,\|\, X_6))$ simply computes the values backwards, and outputs $X_0 \,\|\, X_1$. Construction Ψ_5 is clearly

$(\mathbf{R}^5_{n,n} \to \mathbf{P}_{2n})$-compatible, where we use the notation $\mathbf{R}^k_{n,n}$ to denote the k-fold combination of independent random functions which takes queries of the form (i, x) that are answered by evaluating on x the i-th function.

The following theorem establishes CP-indifferentiability for Ψ_5. We discuss below its consequences, and give a detailed description of our simulation strategy. The full analysis of the simulation strategy – which employes the randomness-mapping technique of [29] – is found in the full version.

Theorem 5 (CP-indifferentiability of Feistel). *Let* $\mathbf{R} = \mathbf{R}^5_{n,n}$ *and* $\mathbf{P} = \mathbf{P}_{2n}$*. Then, there exists a simulator* Sim *(described in Fig. 6) such that for all distinguisher* A *making at most* $q(\lambda)$ *queries,*

$$\mathsf{Adv}^{\mathsf{cpi}[\mathbf{R}\to\mathbf{P}]}_{\Psi_5,\mathsf{Sim},A}(\lambda) \leq \frac{360q(\lambda)^6}{2^{n(\lambda)}} \ . \tag{13}$$

Here, Sim *makes at most* $2q(\lambda)^2$ *queries, and otherwise runs in time polynomial in the number of queries answered, and* n*.* ■

This, together with Theorem 1, gives us immediately the following corollary: Given a keyed function family $\mathsf{F} = (\mathsf{F.Kg}, \mathsf{F.Eval})$, where for all $\lambda \in \mathbb{N}$, $k \in [\mathsf{F.Kg}(1^\lambda)]$, $\mathsf{F.Eval}(1^\lambda, k, \cdot)$ is a function from $n(\lambda)+3$ bits to $n(\lambda)$ bits, interpreted as a function $[5] \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$, then define the keyed function family $\Psi_5[\mathsf{F}] = (\Psi.\mathsf{Kg}, \Psi.\mathsf{Eval})$ obtained by instantiating the round function using F.

Corollary 2. *For any polynomially bounded* $n = \omega(\log \lambda)$*, if* $\mathsf{F} \in \mathsf{UCE}[n + 3, \mathcal{S}^{\star\mathsf{rs}}]$*, then* $\Psi_5[\mathsf{F}] \in \mathsf{psPRP}[2n, \mathcal{S}^{\star\mathsf{rs}}]$*, where* $\star \in \{\mathsf{c}, \mathsf{s}\}$*.* ■

REMARKS. Theorem 5 is interesting in its own right, as part of the line of works on (full-fledged) indifferentiability of Feistel constructions. Coron et al. [21] show that six rounds are necessary for achieving indifferentiability, and proofs of indifferentiability have been given for 14, 10, and 8 rounds, respectively [29,21,22,23]. Thus, our result shows that CP-indifferentiability is a strictly weaker goal in terms of round-complexity of the Feistel construction. (Also for sequential indifferentiability as in [33], six rounds are necessary.) As we will see in the next paragraph, our simulation strategy departs substantially from earlier proofs.

Two obvious problems remain open. First off, we know four rounds are necessary (as they are needed for indistinguishability alone [32]), but we were unable to make any progress on whether CP-sequential indifferentiability (or psPRP security) is achievable. The second is the case of unpredictable sources. We note that a heavily unbalanced Feistel construction (where each round function outputs one bit) would be query extractable, as the input of the round function leaves little uncertainty on the inner state, and the extractor can evaluate the round functions for other rounds to infer the input/output of the construction. Thus, if we could prove CP-indifferentiabilty, we could combine this with Theorem 2. Unfortunately, such a proof appears beyond our current understanding.

```
PROCEDURE Sim(k, X):
1:  if G_k[X] = ⊥ then
2:    if k = 2 then
3:      F^inner(k, X)
4:      foreach (X_1, X_2) ∈ G_1 × {X} do
5:        if (X_1, X_2, 1) ∉ CompletedChains then
6:          X_0 ← F^inner(1, X_1) ⊕ X_2
7:          (X_5, X_6) ← Func(+, X_0||X_1)
8:          C ← (X_1, X_2, 1)
9:          if G_5[X_5] ≠ ⊥ then                    // Immediate Completion
10:           Complete(C, (X_5, X_6))
11:         else                                    // Completion is delayed
12:           X_3 ← F^inner(2, X_2) ⊕ X_1
13:           Chains[3, X_3] ← (5, X_5), Chains[5, X_5] ←∪ {(C, (X_5, X_6))}
14:   elseif k = 4 then
15:     F^inner(k, X)
16:     foreach (X_4, X_5) ∈ {X} × G_5 do
17:       if (X_4, X_5, 4) ∉ CompletedChains then
18:         X_6 ← F^inner(4, X_4) ⊕ X_5
19:         (X_0, X_1) ← Func(-, X_5||X_6)
20:         C ← (X_4, X_5, 4)
21:         if G_1[X_1] ≠ ⊥ then                    // Immediate Completion
22:           Complete(C, (X_0, X_1))
23:         else                                    // Completion is delayed
24:           X_3 ← F^inner(4, X_4) ⊕ X_5
25:           Chains[3, X_3] ← (1, X_1), Chains[1, X_1] ←∪ {(C, (X_0, X_1))}
26:   elseif k ∈ {1, 5} then
27:     F^inner(k, X)
28:     foreach (C, (U, V)) ∈ Chains[k, X] do
29:       if C ∉ CompletedChains then               // Delayed Completion
30:         Complete(C, (U, V))
31:   elseif Chains[3, X] ≠ ⊥ then
32:     Sim(Chains[k, X])
33: return F^inner(k, X)
```

Fig. 6: The code for simulator Sim. Sim has access to the Func oracle and maintains data structures G_k, Chains and CompletedChains as global variables.

Simulator description. We explain now our simulation strategy, which is described formally in Fig. 6. We note that our approach inherits the chain-completion technique from previous proofs, but it will differ substantially in how and when chains are completed.

Recall that in the ideal case, in the first stage of the CP-indifferentiability game, A_1 makes queries to Func implementing a random permutation, and then passes the control of the game to A_2 which interacts with Sim. Our Sim maintains tables G_k for $k \in [5]$ to simulate the round functions. We denote by $G_k[X] = \perp$ that the table entry for X is undefined, and we assume all values

PROCEDURE $\mathsf{F}^{\text{inner}}(i, X_i)$:
34: **if** $G_i[X_i] = \perp$ **then**
35: $G_i[X_i] \leftarrow_{\$} \{0,1\}^n$
36: **return** $G_i[X_i]$

PROCEDURE $\mathsf{ForceVal}(X, Y, l)$:
37: $G_l[X] \leftarrow Y$

PROCEDURE $\mathsf{Complete}(C, (U, V))$:
38: $(X, Y, i) \leftarrow C$
39: **if** $i = 1$ **then**
40: $X_1 \leftarrow X, X_2 \leftarrow Y,\ X_3 \leftarrow \mathsf{F}^{\text{inner}}(2, X_2) \oplus X_1$
41: $(X_5, X_6) \leftarrow (U, V)$
42: $X_4 \leftarrow \mathsf{F}^{\text{inner}}(5, X_5) \oplus X_6$
43: $\mathsf{ForceVal}(X_3, X_4 \oplus X_2, 3)$, $\mathsf{ForceVal}(X_4, X_5 \oplus X_3, 4)$
44: **elseif** $i = 4$ **then**
45: $X_4 \leftarrow X, X_5 \leftarrow Y,\ X_3 \leftarrow \mathsf{F}^{\text{inner}}(4, X_4) \oplus X_5$
46: $(X_0, X_1) \leftarrow (U, V)$
47: $X_2 \leftarrow \mathsf{F}^{\text{inner}}(1, X_1) \oplus X_0$
48: $\mathsf{ForceVal}(X_3, X_4 \oplus X_2, 3)$, $\mathsf{ForceVal}(X_2, X_1 \oplus X_3, 2)$
49: CompletedChains $\overset{\cup}{\leftarrow} \{(X_1, X_2, 1), (X_4, X_5, 4)\}$

Fig. 6: (continued) The code for subroutines used by simulator Sim.

are initially undefined. Also, we refer to a tuple (X_k, X_{k+1}, k) as a *partial chain* where $G_k[X_k] \neq \perp$ and $G_{k+1}[X_{k+1}] \neq \perp$ for $k \in \{1, 4\}$, $X_k, X_{k+1} \in \{0,1\}^n$.

For any query (k, X) by A_2, Sim checks if $G_k[X] = \perp$. If not then the image $G_k[X]$ is returned. Otherwise, depending on the value of k, Sim takes specific steps as shown in Fig. 6. If $k \in \{2, 4\}$ then Sim sets $G_k[X]$ to a uniformly random n-bit string by calling the procedure $\mathsf{F}^{\text{inner}}$. At this point, Sim considers newly formed tuples $(X_1, X_2) \in G_1 \times \{X\}$ (when $k = 2$) and detects partial chains $C = (X_1, X_2, 1)$. The notation $X_1 \in G_1$ is equivalent to $G_1[X_1] \neq \perp$. For every partial chain C that Sim detects, it queries Func on (X_0, X_1) and receives (X_5, X_6) where $X_0 = G_1[X_1] \oplus X_2$. If (X_0, X_1) does not appear in one of the queries/responses by/to A_1 then it is unlikely for A_2 to guess the corresponding (X_5, X_6) pair. Therefore, if $G_5[X_5] \neq \perp$ then Sim assumes that C is a chain that most likely corresponds to a query by A_1. We refer to partial chains that correspond to the queries by A_1 as *relevant* chains. In this case, Sim *immediately* completes C by calling the procedure $\mathsf{Complete}$. C is completed by forcing the values of $G_3[X_3]$ and $G_4[X_4]$ to be consistent with the Func query where $X_3 \leftarrow G_2[X_2] \oplus X_1$ and $X_4 \leftarrow G_5[X_5] \oplus X_6$.

If $G_5[X_5] = \perp$ then either C is not a relevant chain or C is a relevant chain but A_2 has not queried $(5, X_5)$ yet. An aggressive strategy would be to complete C, thereby asking Sim to complete every partial chain ever detected. The resulting simulation strategy will however end up potentially managing an exponential number of partial chains, contradicting our goal of efficient simulation.

Hence, Sim *delays* the completion and only completes C on A_2's query to either $(3, X_3)$ or $(5, X_5)$ where $X_3 = G_2[X_2] \oplus X_1$. The completion is delayed by storing information about X_3 and X_5, that fall on the chain C, in the table Chains. In particular, Sim stores a pointer to $(5, X_5)$ at Chains$[3, X_3]$. The inputs $((X_1, X_2, 1), (X_5, X_6))$ to the Complete call on C are stored in Chains$[5, X_5]$. As many chains can share the same X_5, we allow Chains$[5, X_5]$ to be a set. The idea of delaying the chain completions is unique to our simulation strategy and it translates to an efficient Sim which consistently completes chains in the eyes of A. Sim works symmetrically when $k = 4$.

For queries of the form (k, X) where $k \in \{1, 5\}$, Sim always assigns $G_k[X]$ to a uniform random n-bit string by calling $\mathsf{F}^{\text{inner}}$. Moreover as discussed earlier, X could be on previously detected partial chains whose completion was delayed. Therefore after the assignment, Sim picks up all partial chains C' (if any) stored in Chains$[k, X]$ and completes them. This is where Sim captures a relevant partial chain which was delayed for completion. Finally for queries $(3, X)$, Sim checks if this X was on a partial chain that was detected but not completed. If Chains$[3, X] = \perp$ then Sim assigns $G_3[X]$ a uniform random n-bit string otherwise it follows the pointer to Chains$[3, X]$ to complete the chain X was on. Since Chains$[3, X]$ just stores a tuple (instead of a set) there can be at most one chain C that Chains$[3, X]$ can point to at any time. In the execution, Chains$[3, X]$ can get overwritten which may lead to inconsistencies in chain completions. However, we show that there are no overwrites in either tables G_k or the data-structure Chains, except with negligible probability. This allows Sim to Complete chains consistently in the eyes of A. Furthermore, to avoid completing the same chains again, Sim maintains a set of all CompletedChains and completes any chain if it is not in CompletedChains. A pictorial description of Sim is found in Fig. 7.

6 Ideal-model psPRP Constructions

We discuss two natural approaches to instantiate psPRPs. One is by taking any block cipher, and using its key as a (now public) seed. The second is by using a key-less permutation (e.g., the one within SHA-3), and adding the seed through the Even-Mansour [25] construction. While the purpose of our psPRP framework is to remove ideal-model assumptions, the only obvious way to validate (heuristically) these methods *is* via ideal-model proofs, and this is what we do here. Also, note that such ideal-model proofs are useful since, as in the case of UCE, psPRP security can become a powerful intermediate notion within ideal-model proofs for multi-stage security games [36].

PsPRPs from block ciphers. Given a family of block ciphers $\mathsf{E}_\lambda : \{0,1\}^{s(\lambda)} \times \{0,1\}^\lambda \to \{0,1\}^\lambda$, we consider the construction $\mathsf{F} = (\mathsf{F.Kg}, \mathsf{F.Eval})$, where $\mathsf{F.Kg}(1^\lambda)$ outputs a random $k \leftarrow\!\!\$\, \{0,1\}^{s(\lambda)}$, whereas

$$\mathsf{F.Eval}(1^\lambda, (k, +, x)) = \mathsf{E}(k, x)\ , \quad \mathsf{F.Eval}(1^\lambda, (k, -, y)) = \mathsf{E}^{-1}(k, y)\ . \tag{14}$$

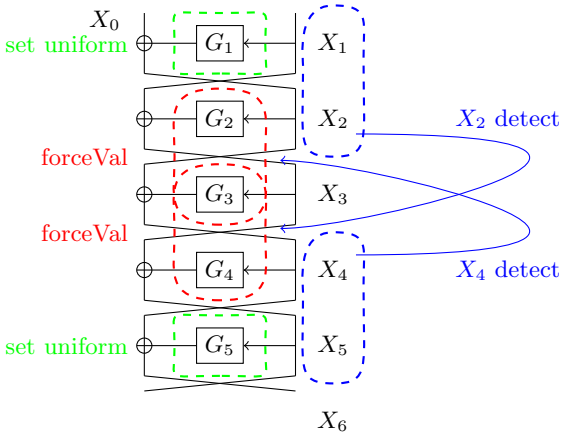

Fig. 7: The 5-round Feistel where Sim sets $G_1[X_1]$ and $G_5[X_5]$ uniformly at random (green). Sim detects chains at either (X_1, X_2) or (X_4, X_5) (blue) and adapts at (X_3, X_4) and (X_2, X_3) respectively (red).

The following theorem establishes its security in the ideal cipher model, that is, we assume (without overloading notation) that all parties (i.e., the source, the distinguisher, and the reset adversary in the proof) are given access to a randomly chosen block cipher E, which is also used within F. We refer the reader to the full version for the proof which closely follows the proof from [8] that a random oracle is UCE-secure.

Theorem 6 (Ideal Cipher as a psPRP). *Let* $\mathbf{P}$ *be the random permutation with input length* $n(\lambda) = \lambda$*. For every source-distinguisher pair* S, D*, where the source* S*, in its first stage, outputs* n *which is at most* $N(\lambda)$ *and makes* q Prim *queries to its oracle, there exists* R *(described in the proof) such that*

$$\mathsf{Adv}^{\mathsf{psPRP}[n]}_{\mathsf{F},S,D}(\lambda) \leq \mathsf{Adv}^{\mathsf{reset}[\mathbf{P}]}_{S,R}(\lambda) + \frac{2qN(\lambda)}{2^{s(\lambda)}} + \frac{2N(\lambda)^2}{2^{s(\lambda)}} \ . \tag{15}$$

In particular, if D *is polynomial time, then so is* R*.* ■

EVEN-MANSOUR. We find it practically valuable to assess whether simple constructions can work. To this end, here, we show that the Even-Mansour construction [25] yields a $\mathsf{psPRP}[\mathcal{S}^{\mathsf{cup}}]$-secure permutation family in the random permutation model. In particular, we assume we are given a family of permutations $\Pi = \{\pi_\lambda\}_{\lambda\in\mathbb{N}}$, where $\pi_\lambda : \{0,1\}^\lambda \to \{0,1\}^\lambda$, and consider the construction $\mathsf{EM} = (\mathsf{EM.Kg}, \mathsf{EM.Eval})$ as defined in Section 4.1. Similar to the above, the following theorem implicitly assumes all parties are given oracle access to a random permutation which is used to sample the permutation inside EM.

Below, we give a few remarks on why the above approach to show $\mathsf{psPRP}[\mathcal{S}^{\mathsf{crs}}]$ security does not extend to the case of Even-Mansour.

Theorem 7 (Even-Mansour as a psPRP). *Let* $\mathbf{P}$ *be the random permutation with input length* $n(\lambda) = \lambda$*. For every source-distinguisher pair* S, D*, where the source* S*, in its first stage, outputs* n *which is at most* $N(\lambda)$ *and where* S *and* D *jointly make at most* q *queries to their oracles, there exists* P *(described in the proof) such that*

$$\mathsf{Adv}^{\mathsf{psPRP}[n]}_{\mathsf{EM},S,D}(\lambda) \leq \mathsf{Adv}^{\mathsf{pred}[\mathbf{P}]}_{S,P}(\lambda) + \frac{3q^2}{2^\lambda} + \frac{2N(\lambda)q^2}{2^\lambda} \ . \tag{16}$$

In particular, if D *runs in polynomial time, then so does* P*.* ■

The proof of Theorem 7 can be found in the full version. It resembles the original indistinguishability proof from [25], which bounds the advantage via the probability of an intersection query, that is, a direct query (by the source or by the distinguisher) to the random permutation that overlaps with one of the queries to the random permutation made internally by oracle $\mathcal{O}$ invoked by the source. Bounding the probability that S makes an intersection query proceeds as in [25] (exploiting lack of knowledge of the seed), whereas bounding the probability that D makes such a query requires a reduction to unpredictability.

Why not reset-secure sources? We would like to extend Theorem 7 to $\mathcal{S}^{\mathsf{crs}}$, as this would provide validation for the assumption from Section 4.1. While we conjecture this to be true, the statement seems to evade a simple proof. The proof approach behind Theorem 6 fails in particular, as it heavily exploits the property that for each distinct seed, the construction F queries a disjoint portion of the domain of the ideal cipher, which is not true for EM.

7 Efficient Garbling from psPRPs

As an application of the psPRP framework, we study the security of the efficient garbling schemes of Bellare, Hoang, Keelveedhi, and Rogaway [10], and in particular, their simplest scheme (called Ga). It follows Yao's general garbling paradigm [40], but proposes a particular gadget to garble individual gates that only relies on evaluating the underlying block cipher on a fixed key. In terms of efficiency, this has been shown to be advantageous, as it avoids higher re-keying costs. However, its security has only been proved in the ideal-cipher model, and recent work by Gueron et al. [28] has debated this. Here, we show that a minor variant of Ga (which still largely benefits from the lack of re-keying) is secure assuming the underlying block cipher is $\mathsf{psPRP}[\mathcal{S}^{\mathsf{sup}}]$-secure. While this assumption is undoubtedly strong, it makes it clear what is expected from the permutation. In particular, the main concern of [28] is the existence of fixed-key distinguishers (as in [30]), but these do not seem to affect psPRP-security, while they may invalidate the permutation being ideal.

Simple circuit description. For representing circuits we adopt the SCD notation of [11]. A circuit is a 6-tuple $f = (n, m, q, W_1, W_2, G)$ where $n \geq 2$ is the number of inputs, $m \geq 1$ is the number of outputs, $q \geq 1$ is the number of gates, and $n+q$ is the number of wires. We let $\mathsf{Inputs} = [1, \ldots, n]$, $\mathsf{Wires} = [1, \ldots, n+q]$,

$\mathsf{OutputWires} = [n+q-m+1,\ldots,n+q]$ and $\mathsf{Gates} = [n+1,\ldots,n+q]$. Then $W_1 : \mathsf{Gates} \to \mathsf{Wires} \setminus \mathsf{OutputWires}$ is a function to identify each gate's first incoming wire and $W_2 : \mathsf{Gates} \to \mathsf{Wires} \setminus \mathsf{OutputWires}$ to identify each gate's second incoming wire. Finally $G : \mathsf{Gates} \times \{0,1\}^2 \to \{0,1\}$ is a function that determines the functionality of each gate i.e. G_g is a table storing the output of gate g with input i and j at $G_g[i,j]$. We require that $W_1(g) < W_2(g) < g$ for all gates g.

Following [10], our definitions will be parameterized by the side information about the circuit obtained from its garbled counterpart. We consider the *topology side information* ϕ_{topo} which maps f to its topology $\phi_{\mathsf{topo}}(f) = (n, m, q, W_1, W_2)$. Another example is ϕ_{xor}, which maps f to a circuit $\phi_{\mathsf{xor}}(f) = (n, m, q, W_1, W_2, G')$ which obscures the functionality of non-xor gates. As shown in [11] and [10], an important property is that ϕ_{topo} and ϕ_{xor} are efficiently invertible, i.e., there exists an efficient algorithm which given $\phi(f)$ and y, outputs (f', x') such that $\phi(f) = \phi(f')$ and $y = \mathsf{ev}(f', x')$.

Garbling schemes and their security. To describe a garbling scheme we use the notation from [11]. A *garbling scheme* is a tuple of algorithms $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. The algorithm Gb is probabilistic and others are deterministic. Gb takes as inputs a circuit $f = (n, m, q, W_1, W_2, G)$ represented in the SCD notation and a security parameter 1^λ and returns a tuple of strings (F, e, d) where F is the garbled circuit, e is the input encoding information and d is the output decoding information. $\mathsf{En}(e, \cdot) : \{0,1\}^n \to \{0,1\}^*$ transforms the n-bit input x to the garbled input X. $\mathsf{Ev}(F, \cdot) : \{0,1\}^* \to \{0,1\}^*$ runs the garbled circuit F on garbled input X and returns the garbled output Y. $\mathsf{De}(d, \cdot) : \{0,1\}^* \to \{0,1\}^m \cup \{\bot\}$ decodes the garbled output Y to return $y \in \{0,1\}^m$. The algorithm ev is the canonical circuit-evaluation function where $\mathsf{ev}(f, x)$ is the m-bit output one gets by feeding x to f. Finally, we require that $\mathcal{G}$ is *correct*, that is, if $f \in \{0,1\}^*$, $\lambda \in \mathbb{N}$, $x \in \{0,1\}^n$ and $(F, e, d) \in [\mathsf{Gb}(1^\lambda, f)]$, then $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = \mathsf{ev}(f, x)$. We require that all algorithms run in time polynomial in the security parameter λ.

In this work, we are only concerned with indistinguishability-based privacy, as defined in Game $\mathsf{PrvInd}^A_{\mathcal{G},\phi}$ in Fig. 8. Since both ϕ_{topo} and ϕ_{xor} are efficiently invertible, [11] show that it is sufficient to focus on this target, since simulation-based security is implied. We say that $\mathcal{G}$ is prvind-*secure over side information function* ϕ if for all PPT adversaries A,

$$\mathsf{Adv}^{\mathsf{prvind}[\phi]}_{\mathcal{G},A}(\lambda) = 2\Pr\left[\mathsf{PrvInd}^A_{\mathcal{G},\phi}(\lambda)\right] - 1 \tag{17}$$

is negligible (in λ). Also, it is not hard to see (by a simple hybrid argument) that it is sufficient to prove this for adversaries which make one single query to their oracle.

Garbling scheme $\mathsf{Ga}[\mathsf{P}]$. Our garbling scheme resembles heavily that of [10]. The only modification is that we assume it uses a function family P meant to be $\mathsf{psPRP}[\mathcal{S}^{\mathsf{sup}}]$-secure (which could be instantiated from a block cipher, by letting the key take the role of the seed.). During the garbling procedure, a fresh seed for P is chosen and made part of the garbled circuit. Clearly, re-keying costs

MAIN $\mathsf{PrvInd}^A_{\mathcal{G},\phi}(1^\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ A^{\mathsf{Garble}}(1^\lambda)$
return $(b' = b)$

PROCEDURE $\mathsf{Garble}(f_0, f_1, x_0, x_1)$
if $\phi(f_0) \neq \phi(f_1)$ **then return** $\perp$
if $\{x_0, x_1\} \not\subseteq \{0,1\}^{f_0.n}$ **then return** $\perp$
if $\mathsf{ev}(f_0, x_0) \neq \mathsf{ev}(f_1, x_1)$ **then return** $\perp$
$(F, e, d) \leftarrow\!\!\$\ \mathsf{Gb}(1^\lambda, f_b)$, $X \leftarrow \mathsf{En}(e, x_b)$
return (F, X, d)

Fig. 8: $\mathsf{PrvInd}^A_{\mathcal{G},\phi}$ Game for $\mathcal{G}$ with adversary A.

PROCEDURE $\mathsf{En}(e, x)$
$(X^0_1, X^1_1, \ldots, X^0_n, X^1_n) \leftarrow e$
$x_1, \ldots, x_n \leftarrow x$
$X \leftarrow (X^{x_1}_1, \ldots, X^{x_n}_n)$
return X

PROCEDURE $\mathsf{Ev}(F, X)$
$(n, m, q, W_1, W_2, T, s) \leftarrow F$
$(X_1, \ldots, X_n) \leftarrow X$

foreach $g \in [n+1, \ldots, n+q]$ **do**
  $w_1 \leftarrow W_1(g), w_2 \leftarrow W_2(g)$
  $\alpha \leftarrow \mathrm{lsb}(X_{w_1})$, $\beta \leftarrow \mathrm{lsb}(X_{w_2})$
  $K \leftarrow X_{w_1} \oplus X_{w_2} \oplus g$
  $X_g \leftarrow T[g, \alpha, \beta] \oplus \mathsf{P.Eval}(s, K) \oplus K$
return $(X_{n+q-m+1}, \ldots, X_{n+q})$

PROCEDURE $\mathsf{De}(d, Y)$
$(d_1, \ldots, d_m) \leftarrow d$
$(Y_1, \ldots, Y_m) \leftarrow Y$
foreach $i \in [1, \ldots, m]$ **do**
  $y_i \leftarrow \mathrm{lsb}(Y_i) \oplus d_i$
return $y \leftarrow y_1, \ldots, y_m$

PROCEDURE $\mathsf{Gb}(1^\lambda, f)$
$s \leftarrow\!\!\$\ \mathsf{P.Kg}(1^\lambda)$
$(n, m, q, A', B', G) \leftarrow f$

foreach $i \in [1, \ldots, n+q]$ **do**
  $t \leftarrow\!\!\$\ \{0,1\}$
  $X^0_i \leftarrow\!\!\$\ \{0,1\}^{k-1} t$
  $X^1_i \leftarrow\!\!\$\ \{0,1\}^{k-1} \bar{t}$

foreach $g \in [n+1, \ldots, n+q]$ **do**
  $w_1 \leftarrow W_1(g), w_2 \leftarrow W_2(g)$
  foreach $(i, j) \in \{0,1\}^2$ **do**
    $A \leftarrow X^i_{w_1}, \alpha \leftarrow \mathrm{lsb}(A)$
    $B \leftarrow X^i_{w_2}, \beta \leftarrow \mathrm{lsb}(B)$
    $K \leftarrow X^i_{w_1} \oplus X^j_{w_2} \oplus g$
    $T[g, \alpha, \beta] \leftarrow \mathsf{P.Eval}(s, K) \oplus K \oplus X^{G_g[i,j]}_g$

$F \leftarrow (n, m, q, W_1, W_2, T, s)$
$e \leftarrow (X^0_1, X^1_1, \ldots, X^0_n, X^1_n)$
$d \leftarrow (\mathrm{lsb}(X^0_{n+1-m+1}), \ldots, \mathrm{lsb}(X^0_{n+q}))$
return (F, e, d)

Fig. 9: Scheme $\mathsf{Ga}[\mathsf{P}]$.

are still largely avoided (especially for large circuits), even though re-keying is necessary when garbling multiple circuits.

Concretely, the garbling scheme $\mathsf{Ga}[\mathsf{P}]$ (Fig. 9) is a tuple of algorithms $\mathsf{Ga}[\mathsf{P}] = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ where $\mathsf{P} \in \mathsf{psPRP}[k(\lambda), \mathcal{S}^{\mathsf{sup}}]$ for some polynomial $k(\lambda)$. Algorithm Gb transforms the input circuit f to a tuple of strings (F, e, d) where the seed s for the permutation P sampled independently for each input f is now part of F. Though the Ga scheme of [10] comes in several variants, where each variant is defined by the dual-key cipher used, we focus on a specific dual-key cipher (namely A1 in [10]) that leads to the most efficient implementation of Ga.

In the following theorem we prove the prvind-security of $\mathsf{Ga}[\mathsf{P}]$ and later discuss about the more efficient schemes GaX and GaXR from [10].

Theorem 8 (Garbling from psPRPs). *Let* $\mathsf{P} \in \mathsf{psPRP}[k(\lambda), \mathcal{S}^{\mathsf{sup}}]$ *then* $\mathsf{Ga}[\mathsf{P}]$ *is* prvind*-secure over* ϕ_{topo}. ■

Proof. Let us assume that $\mathsf{Ga}[\mathsf{P}]$ is not prvind-secure then there exists a PPT adversary A that issues circuits with at most $q(\lambda)$ gates and achieves a non-negligible advantage $\varepsilon(\lambda)$ in the $\mathsf{PrvInd}^{A}_{\mathsf{Ga}[\mathsf{P}],\phi}(\lambda)$ [9] game. Using A we construct a pair (S, D) (Fig. 10) breaking the psPRP security of P, where S is a statistically unpredictable source. Without loss of generality, we can assume that A queries its oracle exactly once.

Let c be the challenge bit in the psPR game for P, and let Perm be the oracle called by S. We allow S to sample the challenge bit b for the $\mathsf{PrvInd}^{A}_{\mathsf{Ga}[\mathsf{P}],\phi}$ game. Further, for syntactic reasons we decompose A into (A_0, A_1) where A_0 on input 1^λ outputs (f_0, x_0, f_1, x_1) (inputs for Garble) and forwards a state st to A_1. The result of Garble i.e. (F, X, d) is forwarded to A_1 to guess the challenge bit b in the $\mathsf{PrvInd}^{A}_{\mathsf{Ga}[\mathsf{P}],\phi}$ game. The source S nearly acts as Gb on input f_b. To satisfy unpredictability, the leakage L must give no information about the queries made by S. Therefore, S refrains from compiling the rows in the garbled table T which can be opened by A. S outputs this partially garbled circuit F^{-} as leakage in addition to (b, d, st). Moreover, since (S, D) must perfectly simulate the $\mathsf{PrvInd}^{A}_{\mathsf{Ga}[\mathsf{P}],\phi}$ game for A, leakage also contains the vector X^{+} which is the set of all visible tokens (one for each wire). Given s and L, D completes the garbled circuit and invokes A_1 with appropriate inputs. D then outputs $b' \oplus b$ where b' was the guess of A in the $\mathsf{PrvInd}^{A}_{\mathsf{Ga}[\mathsf{P}],\phi}$ game.

It is easy to see that when $c = 1$, (S, D) simulate the game $\mathsf{PrvInd}^{A}_{\mathsf{Ga}[\mathsf{P}],\phi}$ for A. Furthermore, when $c = 0$ the leakage L can be transformed to be independent of the bit b by modifying Perm to act like a random function. This allows rows in the garbled table to be independent of the tokens $X_g^{G_g[i,j]}$ which might depend on bit b. Therefore, in this modified game A can do no better than guessing. For a detailed analysis, we direct the reader to the full version.

To prove that S is statistically unpredictable we need to show that any (possibly unbounded) predictor P making at most $p(\lambda)$ number of queries to the oracle Perm is unlikely to predict a query made by S given $L = (F^{-}, X^{+}, b, d, \mathsf{st})$. The idea is to swiftly transition to a game where L is independent of the queries made by S to Perm. This then reduces P to merely guess the queries. To achieve this, we take a similar path as the psPRP game of P ($c = 0$). We transition to a game G_1 where F^{-} is independent of bit b. However, unlike the psPRP case, P and S share the same oracle Perm (to which P can also make inverse queries), and therefore it is non-trivial to argue about the independence of L and queries of S as we desire. Therefore, we make a final transition to a game G_2 where Perm returns random strings for queries by S and refrains from storing any information about the queries made by S. The resulting leakage can be viewed

[9] We drop the subscript topo from ϕ for ease of notation.

SOURCE $S^{\mathsf{Perm}}(1^\lambda)$:
$b \leftarrow\!\!\$\ \{0,1\}$
$(f_0, x_0, f_1, x_1, \mathsf{st}) \leftarrow A_0(1^\lambda)$
$(n, m, q, W_1, W_2) \leftarrow \phi(f_b)$
$G \leftarrow f_b.G$
foreach $i \in [1, \ldots, n+q]$ **do**
  $v_i \leftarrow \mathsf{ev}(f_b, x_b, i)$; $t_i \leftarrow\!\!\$\ \{0,1\}$
  $X_i^{v_i} \leftarrow\!\!\$\ \{0,1\}^{k-1} t_i$
  $X_i^{\overline{v_i}} \leftarrow\!\!\$\ \{0,1\}^{k-1} \overline{t_i}$
foreach $g \in [n+1, \ldots, n+q]$ **do**
  foreach $(i,j) \in \{0,1\}^2$ **do**
    $w_1 \leftarrow W_1(g)$, $w_2 \leftarrow W_2(g)$
    $A \leftarrow X_{w_1}^i$, $\alpha \leftarrow \mathrm{lsb}(A)$
    $B \leftarrow X_{w_2}^j$, $\beta \leftarrow \mathrm{lsb}(B)$
    $K \leftarrow A \oplus B \oplus g$
    if $i \neq v_{w_1} \vee j \neq v_{w_2}$ **then**
      $T[g, \alpha, \beta] \leftarrow \mathsf{Perm}(K) \oplus K \oplus X_g^{G_g[i,j]}$
$F^- \leftarrow (n, m, q, W_1, W_2, T)$
$X^+ \leftarrow (X_1^{v_1}, \ldots, X_{n+q}^{v_{n+q}})$
return $(F^-, X^+, d, b, \mathsf{st})$

DISTINGUISHER $D(1^\lambda, s, L)$:
$(F^-, X^+, d, b, st) \leftarrow L$
$(n, m, q, W_1, W_2, T) \leftarrow F^-$
$X_1, \ldots, X_{n+q} \leftarrow X^+$
for $g \in [n+1, \ldots, n+q]$ **do**
  $w_1 \leftarrow W_1(g)$, $w_2 \leftarrow W_2(g)$
  $\alpha \leftarrow \mathrm{lsb}(X_{w_1})$, $\beta \leftarrow \mathrm{lsb}(X_{w_2})$
  $K \leftarrow X_{w_1} \oplus X_{w_2} \oplus g$
  $T[g, \alpha, \beta] \leftarrow \mathsf{P.Eval}(s, K) \oplus K \oplus X_g$
$F \leftarrow (n, m, q, W_1, W_2, T, s)$
$X \leftarrow (X_1, \ldots, X_n)$
$b' \leftarrow A_1(1^\lambda, st, F, X, d)$
return $(b' \oplus b)$

Fig. 10: (S, D) in the psPR game of P where A_0's inputs are honest.

as being constructed by S without making any queries to Perm. Then we exploit the fact X^+ information theoretically hides X^- and hence queries by S are hidden from any P making only polynomially many queries to Perm. Again we direct the reader to the full version for a rigorous argument. (We also note that this argument is implicitly contained in the original security proof in the ideal-cipher model.) □

Related schemes. Along with Ga, [10] propose another scheme GaX which achieves faster garbling and evaluation times using the free-xor technique [31]. We consider a variant $\mathsf{GaX}[\mathsf{P}]$ of GaX where (like $\mathsf{Ga}[\mathsf{P}]$) the permutation is replaced by a $\mathsf{psPRP}[\mathcal{S}^{\mathsf{sup}}]$-secure permutation P and the seed for P is sampled freshly for every new instantiation of Gb. The security proof of $\mathsf{GaX}[\mathsf{P}]$ almost readily follows from the security proof of GaX from [10] with slight modifications as done in the proof of $\mathsf{Ga}[\mathsf{P}]$.

The third scheme proposed by [10], called GaXR, further improves over GaX in the size of the garbled table due to the use of row reduction technique at the cost of slower garbling and evaluation times. This means that for every gate, GaXR serves only three rows in the garbled table. Here, we note that adapting GaXR to be proved secure under a suitable psPRP assumption does not appear to have a simple and clear solution, and we leave this as an open problem.

Acknowledgments We wish to thank John Retterer-Moore for his involvement in an earlier stage of this project. This research was partially supported by NSF

grants CNS-1423566, CNS-1528178, CNS-1553758 (CAREER), and IIS-152804, and by a Hellman Fellowship.

References

1. Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 531–550. Springer, Heidelberg, August 2013.
2. Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards understanding the known-key security of block ciphers. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 348–366. Springer, Heidelberg, March 2014.
3. Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of keyed sponge constructions using a modular proof approach. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 364–384. Springer, Heidelberg, March 2015.
4. Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX8 and NORX16: Authenticated encryption for low-end systems. Cryptology ePrint Archive, Report 2015/1154, 2015. http://eprint.iacr.org/2015/1154.
5. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.
6. Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, Heidelberg, May 2016.
7. Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, April 2015.
8. Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Heidelberg, August 2013.
9. Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Cryptography from compression functions: The UCE bridge to the ROM. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 169–187. Springer, Heidelberg, August 2014.
10. Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy*, pages 478–492. IEEE Computer Society Press, May 2013.
11. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796. ACM Press, October 2012.
12. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

13. Mihir Bellare and Igors Stepanovs. Point-function obfuscation: A framework and generic constructions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 565–594. Springer, Heidelberg, January 2016.
14. Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Contention in cryptoland: Obfuscation, leakage and UCE. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 542–564. Springer, Heidelberg, January 2016.
15. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, April 2008.
16. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 33–47. Springer, Heidelberg, August 2010.
17. Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 188–205. Springer, Heidelberg, August 2014.
18. Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via UCEs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 122–141. Springer, Heidelberg, December 2014.
19. Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi. A keyed sponge construction with pseudorandomness in the standard model. In *Proceedings of the Third SHA-3 Candidate Conference*, 2012.
20. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.
21. 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, January 2016.
22. Dana Dachman-Soled, Jonathan Katz, and Aishwarya Thiruvengadam. 10-round feistel is indifferentiable from an ideal cipher. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 649–678. Springer, Heidelberg, May 2016.
23. Yuanxi Dai and John P. Steinberger. Indifferentiability of 8-round feistel networks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 95–120. Springer, Heidelberg, August 2016.
24. Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, April 2015.
25. Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–162, 1997.
26. Peter Gazi, Krzysztof Pietrzak, and Stefano Tessaro. The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 368–387. Springer, Heidelberg, August 2015.

27. Peter Gazi and Stefano Tessaro. Secret-key cryptography from ideal primitives: A systematic overview. In *2015 IEEE Information Theory Workshop, ITW 2015, Jerusalem, Israel, April 26 - May 1, 2015*, pages 1–5. IEEE, 2015.
28. Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 567–578. ACM Press, October 2015.
29. Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 89–98. ACM Press, June 2011.
30. Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 315–324. Springer, Heidelberg, December 2007.
31. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Heidelberg, July 2008.
32. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2), 1988.
33. Avradip Mandal, Jacques Patarin, and Yannick Seurin. On the public indifferentiability and correlation intractability of the 6-round Feistel construction. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 285–302. Springer, Heidelberg, March 2012.
34. Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via UCE. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 56–76. Springer, Heidelberg, March 2014.
35. 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*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.
36. Arno Mittelbach. Salvaging indifferentiability in a multi-stage setting. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 603–621. Springer, Heidelberg, May 2014.
37. Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 306–323. Springer, Heidelberg, August 2014.
38. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.
39. Phillip Rogaway and John P. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 220–236. Springer, Heidelberg, April 2008.
40. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.
41. Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 479–508. Springer, Heidelberg, August 2016.