Provably Robust Sponge-Based PRNGs and KDFs

Peter Gaži[1] and Stefano Tessaro[2]

[1] IST Austria
[2] UC Santa Barbara

Abstract. We study the problem of devising provably secure PRNGs with input based on the sponge paradigm. Such constructions are very appealing, as efficient software/hardware implementations of SHA-3 can easily be translated into a PRNG in a nearly black-box way. The only existing sponge-based construction, proposed by Bertoni *et al.* (CHES 2010), fails to achieve the security notion of robustness recently considered by Dodis *et al.* (CCS 2013), for two reasons: (1) The construction is deterministic, and thus there are high-entropy input distributions on which the construction fails to extract random bits, and (2) The construction is not forward secure, and presented solutions aiming at restoring forward security have not been rigorously analyzed.

We propose a *seeded* variant of Bertoni *et al.*'s PRNG with input which we prove secure in the sense of robustness, delivering in particular concrete security bounds. On the way, we make what we believe to be an important conceptual contribution, developing a variant of the security framework of Dodis *et al.* tailored at the ideal permutation model that captures PRNG security in settings where the weakly random inputs are provided from a large class of possible adversarial samplers *which are also allowed to query the random permutation.*

As a further application of our techniques, we also present an efficient sponge-based key-derivation function (which can be instantiated from SHA-3 in a black-box fashion), which we also prove secure when fed with samples from permutation-dependent distributions.

Keywords: PRNGs, sponges, SHA-3, key derivation, weak randomness

1 Introduction

Generating pseudorandom bits is of paramount importance in the design of secure systems – good pseudorandom bits are needed in order for cryptography to be possible. Typically, software-based *pseudorandom number generators* (PRNGs) collect entropy from system events into a so-called *entropy pool*, and then apply cryptographic algorithms (hash functions, block ciphers, PRFs, etc.) to extract pseudorandom bits from this pool. These are also often referred to as PRNGs *with input*, as opposed to classical seed-stretching cryptographic PRGs.

There have been significant standardization efforts in the area of PRNGs [19,1,6], and an attack-centric approach [21,18,30,26,8] has mostly driven their evaluation. Indeed, the development of a comprehensive formal framework to *prove* PRNG security has been a slower process, mostly due to the complexity of the desirable security goals. First models [21,13,5] only gave partial coverage of the security desiderata. For instance, Barak and Halevi [5] introduced a strong notion of PRNG robustness, but their model could not capture the ability of a PRNG to collect randomness at a low rate. Two recent works [15,17] considerably improved this state of affairs with a comprehensive security framework for PRNG robustness whose inputs are adversarially generated (under some weak entropy constraints). The framework of [15] was recently applied to the study of the Intel on-chip PRNG by Shrimpton and Terashima [29].

This paper continues the investigation of good candidate constructions for PRNGs with inputs which are both practical and provably secure. In particular, we revisit the question of building PRNGs from permutations, inspired by recent sponge-based designs [10,31]. We provide variants of these designs which are provably robust in the framework of [15]. On the way, we also extend the framework of [15] to properly deal with security proofs in ideal models (e.g. when given a random permutation), in particular considering PRNG inputs sampled by adversaries which can make queries to the permutation.

Overall, this paper contributes to the development of a better understanding of sponge-based constructs when processing weakly random inputs. As a further testament of this, we apply our techniques to analyze key-derivation functions using sponge-based hash functions, like SHA-3.

Sponge-based PRNGs. SHA-3 relies on the elegant sponge paradigm by Bertoni, Daemen, Peeters, and van Assche [9]. Beyond hash functions, sponges have been used to build several cryptographic objects. In particular, in later work [10], the same authors put forward a sponge-based design of a PRNG with input. It uses an efficiently computable (and invertible) permutation π, mapping n-bit strings to n-bit strings, and maintains an n-bit state, which is initially set to $S_0 \leftarrow 0^n$. Then, two types of operations can be alternated (for additional parameters $r \leq n$, and $c = n - r$, the latter being referred to as the *capacity*):

- State refresh. Weakly random material (e.g., resulting from the measurement of system events) can be added r-bit at a time. Concretely, given an r-bit string I_i of weakly random bits, the state is refreshed to

$$S_i \leftarrow \pi(S_{i-1} \oplus (I_i \,\|\, 0^c)) \ .$$

- Random-bit generation. Given the current state S_i, we can extract r bits of randomness by outputting $S_i[1 \ldots r]$, and updating the state as $S_{i+1} \leftarrow \pi(S_i)$. This process can be repeated multiple times to obtain as many bits as necessary.

This construction is very attractive. First off, it is remarkably simple. Second, it resembles the structure of the SHA-3/KECCAK hash function, and thus efficient implementations of this PRNG are possible with the advent of more and more optimized SHA-3 implementations in both software and hardware. In fact, recent work by Van Herrewege and Verbauwhede [31] has already empirically validated the practicality of the design. Also, the permutation π does not need to be the KECCAK permutation – one could for example use AES on a fixed key.

PRNG SECURITY. Of course, we would like the simplicity of this construction to be also backed by strong security guarantees. The minimum security requirement is that whenever a PRNG has accumulated sufficient entropy material, the output bits are indistinguishable from random. The original security analysis of [10] proves this (somewhat indirectly) by showing that the above construction is indifferentiable [23] from a "generalized random oracle" which takes a sequence of inputs $I_1, I_2, \ldots$ through refresh operations, and when asked to produce a certain output after k inputs have been processed, it simply applies a random function to the concatenation of $I_1, I_2, \ldots, I_k$. This definition departs substantially from the literature on PRNG robustness, and only provides minimal security – for example, it does not cover any form of state compromise.

In contrast, here we call for a provably-secure sponge-based PRNG construction which is *robust* in the sense of [15]. However, there are two reasons why the construction, as presented above, is not robust.

1) NO FORWARD SECRECY. As already recognized in [10], the above PRNG is not forward secure – in particular, learning the state S just after some pseudorandom bits have been output allows to distinguish them from random ones by just computing $\pi^{-1}(S)$. The authors suggest a countermeasure to this: simply zeroing the upper r bits of the input to π before computing the final state, possibly multiple times if r is small. More formally, given the state S'_k obtained after outputting enough pseudorandom bits, and applying π, we compute $S_k, S'_{k+1}, S_{k+1}, \ldots, S'_{k+t}, S_{k+t}$ as

$$S'_{i+1} \leftarrow \pi(S_i) \ ,$$

for $i = k, \ldots, k+t-1$, where S_i is obtained from S'_i by setting the first r bits to 0. This appears to prevent obvious attacks, and to make the construction more secure as t increases, but no formal validation is provided in [10].

In particular, note that the final state S_{k+t} is *not* random, as its first r bits are all 0. Robustness demands that we obtain random bits from S_{k+t} even when no additional entropy is added – unfortunately we cannot just proceed as above, since this will result in outputting r zero bits. (Also note that applying π also does not make the state random, since π is efficiently invertible.) This indicates that a further modification is needed.

2) LACK OF A SEED. The above sponge-based PRNG is unseeded: This allows for high min-entropy distributions (only short of one bit from maximal entropy) for which the generated bits are not uniform. For example, consider $I = (I_1, \ldots, I_k)$, where each I_j is an r-bit string, and such that I is uniformly distributed under the sole constraint that the first bit of the state S_k obtained after injecting all k blocks $I_1, \ldots, I_k$ into the state is always 0. Then, we can never expect the construction to provide pseudorandom bits under such inputs.

One could restrict the focus to "special" distributions as done in [5], arguing nothing like the above would arise in practice. As discussed in [15], however, arguing which sources are possible is difficult, and following the traditional cryptographic spirit, it would be highly desirable to reduce assumptions on the input distributions, which ideally should be *adversarially generated*, at the cost of introducing a (short) random seed which is independent of the distribution.

We note that the above distribution would also invalidate the weak security expectation from [10]. However, their treatment bypasses this problem by employing the random permutation model, where effectively the randomly chosen permutation acts as a seed, *independent of the input distribution.* We believe however this approach (which is not unique to [10]) to be problematic – the random permutation model is only used *as a tool in the security proof* due to the lack of standard-model assumptions under which the PRNG can be proved secure. Yet, in instantiations, the permutation is fixed. In contrast, a PRNG seed is an *actual short string which can and should be actually randomly chosen.*

Our results. We propose and analyze a new sponge-based seeded construction of a PRNG with input (inspired by the one of [10]) which we prove robust. To this end, we use an extension of the framework of [15] tailored at the ideal-permutation model, and dealing in particular with inputs that are generated by adversarial samplers that can query the permutation. The construction (denoted **SPRG**) uses a seed seed, consisting of s r-bit strings $\mathsf{seed}_0, \ldots, \mathsf{seed}_{s-1}$ (s is not meant to be too large here, not more than 2 or 3 in actual deployment). Then, the construction allows to interleave two operations:

- State refresh. The construction here keeps a state $S_i \in \{0,1\}^n$ and a counter $j \in \{0, 1, \ldots, s-1\}$. Given a string I_i of r weakly random bits, the state is refreshed to
$$S_{i+1} \leftarrow \pi(S_i \oplus (I_i \oplus \mathsf{seed}_j) \,\|\, 0^c) \ ,$$
and j is set to $j + 1 \bmod s$.
- Random-bit generation. Given the current state S_i, we can extract r bits of randomness by computing $S_{i+1} \leftarrow \pi(S_i)$, and outputting the first r bits of S_{i+1}. (This process can be repeated multiple times to obtain as many bits as necessary.) When done, we refresh the state by repetitively zeroing its first r bits and applying π, as described above. (How many times we do this is given by a second parameter – t – which ultimately affects the security of the PRNG.)

For a sketch of **SPRG** see Fig. 5. Thus, the main difference over the PRNG of [10] are (1) The use of a seed, (2) The zeroing countermeasure discussed above, and (3) An additional call to π before outputting random bits. In particular, note that **SPRG** still follows the sponge principle, and in fact (while this may not be the most efficient implementation), can be realized from a sponge hash function (e.g., SHA-3) in an entirely black-box way.[3]

In our proof of security, the permutation is randomly chosen, and both the attacker *and* an adversarial sampler of the PRNG inputs have oracle access to it.

[3] Zeroing the upper r bits when refreshing the state after PRNG output can be done by outputting the top r-bit part to be zeroed, and adding it back in.

In fact, an important contribution of our work is that of introducing a security framework for PRNG security based on [15,29] for the ideal permutation model, and we see this as a step of independent interest towards a proper treatment of ideal-model security for PRNG constructions. As a word of warning, we stress that our proofs consider a *restricted* class of permutation-dependent distribution samplers, where the restriction is in terms of imposing an unpredictability constraint which must hold even under (partial) exposure of some (but not all) of the sampler's queries. While our notion is general enough to generalize previous oracle-free samplers and to encompass non-trivial examples (in particular making seedless extraction impossible, which is what we want for the model to be meaningful), we see potential for future research in relaxing this requirement.

Sponge-based key derivation. Our techniques can be used to immediately obtain provable security guarantees for sponge-based key-derivation functions (KDFs). (See Section 6.) While the security of sponge-based KDFs already follows from the original proof of [9], our result will be stronger in that it will also hold for larger classes of permutation-dependent sources. We elaborate on this point a bit further down in the last paragraph of the introduction, mentioning further related work.

Our techniques. Our analysis follows from two main results, of independent interest, which we briefly outline here. Both results are obtained using Patarin's H-coefficient method, as reviewed in Section 2.

The first result – which we refer to as the *extraction lemma* – deals with the ability of extracting keys from weak sources using sponges. In particular, we consider the seeded construction **Sp** which starting from some initial state $S_0 = \mathsf{IV}$, and obtaining r-bit blocks $I_1, \ldots, I_k$ from a weak random source, and a seed $\mathsf{seed} = (\mathsf{seed}_0, \ldots, \mathsf{seed}_{s-1})$, iteratively computes $S_1, \ldots, S_k$ as

$$S_i \leftarrow \pi(S_{i-1} \oplus (I_i \oplus \mathsf{seed}_j) \,\|\, 0^c) \ ,$$

where j is incremented modulo s after each iteration. Ideally, we want to prove that if $I_1, \ldots, I_k$ has high min-entropy h, then the output S_k is random, as long as the adversary (who can see the seed and choose the IV) cannot query the permutation more than (roughly) 2^h times.[4] Note that this cannot be true in general – take e.g. $k = 1$, and even if I_1 is uniformly random, one single inversion query $\pi^{-1}(S_1)$ is enough to distinguish S_1 from a random string, as in the former case the lower c bits will equal those of the IV. Still, we will be able to prove that this attack is the only way to distinguish – roughly, we will prove that S_k *is uniform* as long as the adversary does not query $\pi^{-1}(S_k)$ when given a random S_k, except with negligible probability. This will be good enough for key-derivation applications, where we will need this result for specific adversaries for

[4] One may hope to prove a result which is independent of the number of queries, akin to [14], as after all this structure resembles that of CBC. Yet, we will need to restrict the number of queries for the overall security to hold, and given this, we can expect better extraction performance – in particular, the output can be uniform for $h \ll n$, whereas $h \geq n$ would be necessary if we wanted an unrestricted result.

which querying $\pi^{-1}(S_k)$ will correspond to querying the *secret key* for an already secure construction. In fact, we believe the approach of showing good extraction properties for restricted adversaries to be novel for ideal-model analyses, and of potential wider appeal. (A moral analogue of this in the standard-model is the work of Barak *et al.* on application-specific entropy-loss reduction for the leftover-hash lemma [4].)

We note that the extraction lemma is even more general – we will consider a generalized extraction game where an adversary can adaptively select a subset of samples from an (also adversarial) distribution sampler with the guarantee of having sufficient min-entropy. We also note that at the technical level this result is inspired by recent analyses of key absorption stages within sponge-based PRFs using key-prepending [3,20]. Nonetheless, these works only considered the case of uniform keys, and not permutation-dependent weakly-random inputs.

Another component of possibly independent interest studies the security of the step generating the actual random bits, when initialized with a state of sufficient pseudorandomness. This result will show that security increases with the number t of zeroing steps applied to the state, i.e., the construction is secure as long as the adversary makes less than 2^{rt} queries.

RELATED WORK ON ORACLE DEPENDENCE. As shown in [28], indifferentiability does not have any implications on multi-stage games such as robustness for permutation-dependent distributions. Indeed, [28] was also the first work (to the best our knowledge) to explicitly consider primitive-dependent samplers, in the context of deterministic and hedged encryption. These results were further extended by a recent notable work of Mittelbach [25], who provided general conditions under which indifferentiability can be used in multi-stage settings.

We note that Mittelbach's techniques can be used to prove that some indifferentiable hash constructions are good extractors. However, this does not help us in proving the extraction lemma, as the construction for which we prove the lemma is not indifferentiable to start with, and thus the result fails. There is hope however that Mittlebach's technique could help us in proving our KDF result of Section 6 via the indifferentiability proof for sponges [9] possibly for an even larger class of permutation dependent samplers. We are not sure whether this is the case, and even if possible, what the quantitative implications would be – Mittelbach results are not formulated in the framework of sponges. In contrast, here we obtain our result as a direct corollary of our extraction lemma.

We also note that oracle-dependence was further considered in other multi-stage settings, for instance for related-key security [2]. Also, oracle-dependence can technically be seen as a form of seed-dependence, as considered e.g. in [16], but we are not aware of any of their techniques finding applications in our work.

2 Preliminaries

BASIC NOTATION. We denote $[n] := \{1, \ldots, n\}$. For a finite set $\mathcal{S}$ (e.g., $\mathcal{S} = \{0,1\}$), we let $\mathcal{S}^n$ and $\mathcal{S}^*$ be the sets of sequences of elements of $\mathcal{S}$ of length n and of arbitrary length, respectively. We denote by $S[i]$ the i-th element of $S \in \mathcal{S}^n$

for all $i \in [n]$. Similarly, we denote by $S[i \dots j]$, for every $1 \leq i \leq j \leq n$, the subsequence consisting of $S[i], S[i+1], \dots, S[j]$, with the convention that $S[i \dots i] = S[i]$. $S_1 \,\|\, S_2$ denotes the concatenation of two sequences $S_1, S_2 \in \mathcal{S}^*$, and if $\mathcal{S}_1, \mathcal{S}_2$ are two subsets of $\mathcal{S}^*$, we denote by $\mathcal{S}_1 \,\|\, \mathcal{S}_2$ the set $\{S_1 \,\|\, S_2 \,:\, S_1 \in \mathcal{S}_1, S_2 \in \mathcal{S}_2\}$. Moreover, for a single-element set $\mathcal{S}_1 = \{X\}$ we simplify the notation by writing $X \,\|\, \mathcal{S}_2$ instead of $\{X\} \,\|\, \mathcal{S}_2$. We let $\mathsf{Perms}(n)$ be the set of all permutations on $\{0,1\}^n$. We denote by $X \xleftarrow{\$} \mathcal{X}$ the process of sampling the value X uniformly at random from a set $\mathcal{X}$. For a bitstring $X \in \{0,1\}^*$, we denote by $X_1, \dots, X_\ell \xleftarrow{r} X$ parsing it into ℓ r-bit blocks, using some fixed padding method. The distance of two discrete random variables X and Y over a set $\mathcal{X}$ is defined as $\mathbf{SD}(X,Y) = \frac{1}{2}\sum_{x\in\mathcal{X}} |\mathsf{Pr}[X = x] - \mathsf{Pr}[Y = x]|$. Finally, recall that the min-entropy $\mathbf{H}_\infty(X)$ of a random variable X with range $\mathcal{X}$ is defined as $-\log\left(\max_{x\in\mathcal{X}} \mathsf{Pr}[X = x]\right)$.

<u>Game-based definitions.</u> We use the game-playing formalism in the spirit of [7]. For a game G, we denote by $\mathsf{G}(\mathcal{A}) \Rightarrow 1$ the event that after an adversary $\mathcal{A}$ plays this game, the game outputs the bit 1. Similarly, $\mathsf{G}(\mathcal{A}) \rightarrow 1$ denotes the event that the output of the adversary $\mathcal{A}$ itself is 1.

<u>Ideal permutation model.</u> We perform our analysis in the *ideal permutation model (IPM)*, where each party has oracle access to a public, uniformly random permutation π selected at the beginning of any security experiment. For any algorithm A, we denote by A^π (or $A[\pi]$) that it has access to an oracle permutation π, which can be queried in *both* the forward and backward direction. In the game descriptions below, we sometimes explicitly mention the availability of π to the adversary as oracles π and π^{-1} for forward and backward queries, respectively. We define a natural distinguishing metric for random variables in the IPM. Given two distributions D_0 and D_1, possibly dependent on the random permutation $\pi \xleftarrow{\$} \mathsf{Perms}(n)$, and an adversary $\mathcal{A}$ querying π, we denote

$$\mathsf{Adv}^{\mathsf{dist}}_{\mathcal{A}}(D_0, D_1) = \mathsf{Pr}\left[X \xleftarrow{\$} D_0^\pi \,:\, \mathcal{A}^\pi(X) \Rightarrow 1\right] - \mathsf{Pr}\left[X \xleftarrow{\$} D_1^\pi \,:\, \mathcal{A}^\pi(X) \Rightarrow 1\right] \,.$$

We call $\mathcal{A}$ a q_π-adversary if it asks q_π queries to π.

<u>PRNGs with input.</u> We use the framework of [15] where a *PRNG with input* is defined as a triple of algorithms $\mathbf{G} = (\mathsf{setup}, \mathsf{refresh}, \mathsf{next})$ parametrized by integers $n, r \in \mathbb{N}$, such that:

- setup is a probabilistic algorithm that outputs a public parameter seed;
- $\mathsf{refresh}$ is a deterministic algorithm that takes seed, a state $S \in \{0,1\}^n$, and an input $I \in \{0,1\}^*$, and outputs a new state $S' \leftarrow \mathsf{refresh}(\mathsf{seed}, S, I) \in \{0,1\}^n$;
- next is a deterministic algorithm that takes seed and a state $S \in \{0,1\}^n$, and outputs a pair $(S', R) \leftarrow \mathsf{next}(\mathsf{seed}, S) \in \{0,1\}^n \times \{0,1\}^r$ where S' is the new state and R is the PRNG output.

The parameters n, r denote the state length and output length, respectively. Note that in contrast to [15], we do not restrict the length of the input I to $\mathsf{refresh}$. In this paper, we repeatedly use the term "PRNG" to denote a PRNG with input in the sense of the above definition.

THE H-COEFFICIENT METHOD. We give the basic theorem underlying the H-Coefficient method [27], as recently revisited by Chen and Steinberger [11].

Let $\mathcal{A}$ be a deterministic, computationally unbounded adversary trying to distinguish two experiments that we call real, respectively ideal, with respective probability measures $\mathsf{Pr}^{\mathsf{real}}$ and $\mathsf{Pr}^{\mathsf{ideal}}$. Let $\mathsf{T}_{\mathsf{real}}$ (resp. $\mathsf{T}_{\mathsf{ideal}}$) denote the random variable of the transcript of the real (resp. ideal) experiment that contains everything that the adversary was able to observe during the experiment. Let $\mathsf{GOOD} \cup \mathsf{BAD}$ be a partition of all valid transcripts into two sets – we refer to the elements of these sets as good and bad transcripts, respectively. Then we have:

Theorem 1 (H-Coefficient Method). *Let $\delta, \varepsilon \in [0,1]$ be such that:*

(a) $\mathsf{Pr}[\mathsf{T}_{\mathsf{ideal}} \in \mathsf{BAD}] \leq \delta$.
(b) *For all* $\tau \in \mathsf{GOOD}$, $\mathsf{Pr}[\mathsf{T}_{\mathsf{real}} = \tau]/\mathsf{Pr}[\mathsf{T}_{\mathsf{ideal}} = \tau] \geq 1 - \varepsilon$.

Then $\left|\mathsf{Pr}^{\mathsf{ideal}}(\mathcal{A} \Rightarrow 1) - \mathsf{Pr}^{\mathsf{real}}(\mathcal{A} \Rightarrow 1)\right| \leq \mathbf{SD}(\mathsf{T}_{\mathsf{real}}, \mathsf{T}_{\mathsf{ideal}}) \leq \varepsilon + \delta$.

3 PRNG Security in the IPM

In this section, we adapt the notions of robustness, recovering security, and preserving security for PRNGs [15] to the ideal permutation model and to cover sponge-based designs.[5] This will require several extensions.

First, we adjust for the presence of the permutation oracle π available to all parties. In particular, we need a notion of a legitimate distribution sampler that can query the permutation. Second, our definitions take into account that the state of the sponge-based PRNG at some important points (e.g. after extraction) is not required to be close to a uniformly random string, but rather to a uniform element of $0^r \,\|\, \{0,1\}^c$ instead. Note that this is an instance of a more general issue raised already in [29], as we discuss below.

We then proceed by proving that these modified notions still maintain the useful property shown in [15,29], namely that the combination of recovering security and preserving security still implies the robustness of the PRNG.

3.1 Oracle-dependent randomness and distribution samplers

This section discusses the issue of generating randomness in a model where a randomly sampled permutation $\pi \xleftarrow{\$} \mathsf{Perms}(n)$ is available to all parties. We give a formal definition of adversarial distribution samplers to be used within the PRNG security notions formalized further below.

For our purposes, an (oracle-dependent) *source* $\mathcal{S} = \mathcal{S}^{\pi}$ is an input-less randomized oracle algorithm which makes queries to π and outputs a string X. The *range* of $\mathcal{S}$, denoted $[\mathcal{S}]$, is the set of values x output by $\mathcal{S}^{\pi}$ with positive probability, where the probability is taken over the choice of π and the internal random coins of $\mathcal{S}$.

[5] It is straightforward to extend our treatment to any ideal primitive, rather than just a random permutation – we dispense with doing so for ease of notation.

<u>Distribution samplers.</u> We extend the paradigm of (adversarial) *distribution samplers* considered in [15] to allow for oracle queries to a permutation oracle $\pi \xleftarrow{\$} \mathsf{Perms}(n)$.[6] Recall that in the original formalization, a distribution sampler $\mathcal{D}$ is a randomized stateful algorithm which, at every round, outputs a triple (I_i, γ_i, z_i), where z_i is auxiliary information, I_i is a string, and γ_i is an entropy estimate. In order for such sampler to be legitimate, for every i (up to a certain bound $q_{\mathcal{D}}$), given I_j for every $j \neq i$, as well as $(z_1, \gamma_1), \ldots, (z_{q_{\mathcal{D}}}, \gamma_{q_{\mathcal{D}}})$, it must be hard to predict I_i with probability better than $2^{-\gamma_i}$, in a *worst-case sense* over the choice of I_j for $j \neq i$ and $(z_1, \gamma_1), \ldots, (z_{q_{\mathcal{D}}}, \gamma_{q_{\mathcal{D}}})$.

Extending this worst-case requirement will need some care. To facilitiate this, we consider a specific class of oracle-dependent distribution samplers, which explicitly separate the process of sampling the auxiliary information from the processes of sampling the I_i values. Formally, we achieve this by explicitly requiring that $\mathcal{D}$ outputs (the description of) a source $\mathcal{S}_i$, rather than a value I_i, and the actual value I_i is sampled by running this $\mathcal{S}_i$ once with fresh random coins.

Definition 2 (Distribution samplers). *A* Q-distribution sampler *is a randomized stateful oracle algorithm* $\mathcal{D}$ *which operates as follows:*

- *It takes as input a* state σ_{i-1} *(the initial state is* $\sigma_0 = \bot$*).*
- *On input* σ_{i-1}*,* $\mathcal{D}^\pi(\sigma_{i-1})$ *outputs a tuple* $(\sigma_i, \mathcal{S}_i, \gamma_i, z_i)$*, where* σ_i *is a new state,* z_i *is the auxiliary information,* γ_i *is an entropy estimation, and* $\mathcal{S}_i$ *is a source with range* $[\mathcal{S}_i] \subseteq \{0,1\}^{\ell_i}$ *for some* $\ell_i \geq 1$*. Then, we run* $I_i \xleftarrow{\$} \mathcal{S}_i^\pi$ *to sample the actual value.*
- *When run for* $q_{\mathcal{D}}$ *times, the overall number of queries made by* $\mathcal{D}$ *and* $\mathcal{S}_1, \ldots, \mathcal{S}_{q_{\mathcal{D}}}$ *is at most* $Q(q_{\mathcal{D}})$*. If* $Q = 0$*, then* $\mathcal{D}$ *is called* oracle independent.

We often abuse notation, and compactly denote by $(\sigma_i, I_i, \gamma_i, z_i) \xleftarrow{\$} \mathcal{D}^\pi(\sigma_{i-1})$ *the <u>overall</u> process of running* $\mathcal{D}$ and *the generated source* $\mathcal{S}_i$ *to jointly produce* $(\sigma_i, I_i, \gamma_i, z_i)$*.*

Also we will simply refer to $\mathcal{D}$ as a *distribution sampler*, omitting Q, when the latter is not relevant to the context. Finally, note that in contrast to [15], we consider a relaxed notion where the outputs I_i can be arbitrarily long strings, and are not necessarily fixed length. Still, we assume that the lengths $\ell_1, \ell_2, \ldots$ are a-priori fixed parameters of the samplers, and cannot be chosen dynamically.

We note that this definition appears to exhibit some degree of redundancy. In particular, it seems that without loss of generality one can simply assume that the generated $\mathcal{S}_i$ outputs a fixed value. (Note that $\mathcal{S}_i$ can be chosen itself from a distribution.) However, this separation will be convenient in defining our legitimacy notion for such samplers, as we will distinguish between permutation queries made by $\mathcal{S}_i$, and other permutation queries made by $\mathcal{D}$ (and $\mathcal{S}_j$ for $j \neq i$).

[6] We present the notions here for this specialized case, but needless to say, they extend naturally to other types of randomized oracles, such as random oracles or ideal ciphers.

Game $\mathsf{GLEG}_{q_{\mathcal{D}},i^*}(\mathcal{A}, D)$:

1. Sample $\pi \xleftarrow{\$} \mathsf{Perms}(n)$
2. Run $\mathcal{D}^\pi$ $q_{\mathcal{D}}$ rounds, producing outputs $(\gamma_1, z_1), \ldots, (\gamma_{q_{\mathcal{D}}}, z_{q_{\mathcal{D}}})$, as well as $I_1, \ldots, I_{q_{\mathcal{D}}}$. This in particular entails sampling sources $\mathcal{S}_1, \ldots, \mathcal{S}_{q_{\mathcal{D}}}$, and sampling $I_1, \ldots, I_{q_{\mathcal{D}}}$ from them (recall that each $\mathcal{S}_i$ can query π). Let $\mathcal{Q}_{\mathcal{D}}$ be the set of all input-output pairs of permutation queries made by $\mathcal{D}$ and by $\mathcal{S}_j$ (for $j \neq i^*$) in this process. (The queries made by $\mathcal{S}_{i*}$ are omitted from $\mathcal{Q}_{\mathcal{D}}$.)
3. Run $\mathcal{A}^\pi$ on input $(\gamma_j, z_j)_{j \in [q_{\mathcal{D}}]}$ and $(I_j)_{j \in [q_{\mathcal{D}}] \setminus \{i^*\}}$, and let $V_{\mathcal{A}}$ be $\mathcal{A}$'s final output.
4. The game then outputs $((I_1, \gamma_1, z_1), \ldots, (I_{q_{\mathcal{D}}}, \gamma_{q_{\mathcal{D}}}, z_{q_{\mathcal{D}}}), V_{\mathcal{A}}, Q_{\mathcal{D}})$

Fig. 1. Definition of the game $\mathsf{GLEG}_{q_{\mathcal{D}},i^*}(\mathcal{A}, \mathcal{D})$.

<u>Legitimate distribution samplers.</u> Intuitively, we want to say that once a source $\mathcal{S}_i$ is output with entropy estimate γ_i, then its output has min-entropy γ_i conditioned on everything we have seen so far. However, due to the availability of the oracle π, which is queried by $\mathcal{D}$, by $\mathcal{S}_i$, and by a potential observer attempting to predict the output of $\mathcal{S}_i$, this is somewhat tricky to formalize.

To this end, let $\mathcal{D}$ be a distribution sampler, $\mathcal{A}$ an adversary, and fix $i^* \in [q_{\mathcal{D}}]$, and consider the game $\mathsf{GLEG}_{q_{\mathcal{D}},i^*}(\mathcal{A}, \mathcal{D})$ given in Figure 1. Here, the adversary is given I_j for $j \neq i^*$ and $(z_1, \gamma_1), \ldots, (z_{q_{\mathcal{D}}}, \gamma_{q_{\mathcal{D}}})$, and can make some permutation queries. Then, at the end, the game outputs the combination of $(z_1, \gamma_1, I_1), \ldots, (z_{q_{\mathcal{D}}}, \gamma_{q_{\mathcal{D}}}, I_{q_{\mathcal{D}}})$, the adversary's output, and a transcript of all permutation queries made by (1) $\mathcal{D}$, and (2) $\mathcal{S}_j$ for $j \neq i^*$. We ask that in the *worst case*, the value I_{i^*} cannot be predicted with advantage better than $2^{-\gamma_{i^*}}$ given everything else in the output of the game. Formally:

Definition 3 (Legitimate distribution sampler). *We say that a distribution sampler $\mathcal{D}$ is $(q_{\mathcal{D}}, q_\pi)$-*legitimate*, if for every adversary $\mathcal{A}$ making q_π queries and every $i^* \in [q_{\mathcal{D}}]$, and for any possible values $(I_j)_{j \neq i^*}$, $(\gamma_1, z_1), \ldots, (\gamma_{q_{\mathcal{D}}}, z_{q_{\mathcal{D}}})$, $V_{\mathcal{A}}$, $\mathcal{Q}_{\mathcal{D}}$ potentially output by the game $\mathsf{GLEG}_{q_{\mathcal{D}},i^*}(\mathcal{A}, \mathcal{D})$ with positive probability,*

$$\mathsf{Pr}\left[I_{i^*} = x \,\middle|\, (I_j)_{j \neq i^*}, (\gamma_1, z_1), \ldots, (\gamma_{q_{\mathcal{D}}}, z_{q_{\mathcal{D}}}), V_{\mathcal{A}}, \mathcal{Q}_{\mathcal{D}}\right] \leq 2^{-\gamma_{i^*}} \tag{1}$$

for all $x \in \{0,1\}^{\ell_{i^*}}$, where the probability is conditioned on these particular values being output by the game.

Note that the unpredictability of I_{i^*} is due to what is *not* revealed, including the oracle queries made by $\mathcal{S}_{i^*}$, and the internal random coins of $\mathcal{S}_{i^*}$ *and* $\mathcal{D}$. For instance, for oracle-independent distribution samplers (which we can think of as outputting "constant" sources), our notion of legitimacy is equivalent to the definition of [15]. We show a more interesting example next.

<u>An example: Permutation-based randomness extraction.</u> Consider the simple construction $\mathsf{H}^\pi : \{0,1\}^n \to \{0,1\}^{n/2}$ which on input X outputs the first

$n/2$ bits of $\pi(X)$. It is not hard to prove that if X is an n-bit random variable with high min-entropy k, i.e., $\Pr[\mathsf{X} = X] \leq 2^{-k}$ for all $X \in \{0,1\}^n$, and $\mathsf{U}_{n/2}$ is uniform over the $(n/2)$-bit strings, then for all adversaries $\mathcal{A}$ making q_π queries,

$$\mathsf{Adv}^{\mathsf{dist}}_{\mathcal{A}}(\mathsf{H}^\pi(\mathsf{X}), \mathsf{U}_{n/2}) \leq \mathcal{O}\left(\frac{q_\pi}{2^{n/2}}\right) + \frac{q_\pi}{2^k} \; . \tag{2}$$

The proof (which we omit) would simply go by saying that as long as the attacker does not query X to π (on which it has k bit of uncertainty), or queries $\pi(\mathsf{X})$ to π^{-1} (on which it has only $n/2$ bits of uncertainty), the output looks sufficiently close to uniform (with a tiny bias due to the gathered information about π via $\mathcal{A}$'s direct queries).

Now, let us consider a simple distribution sampler $\mathcal{D}$ which does the following – at every round, regardless of this input, it always outputs the following source $\mathcal{S} = \mathcal{S}^\pi$, as well as $\gamma = n - 1$, and $z = \perp$. The source $\mathcal{S}$ does the following: It queries random n-bit strings X_i to π, until the first bit of $\pi(X_i)$ is 0, and then outputs X_i. It is not hard to show that for any $q_{\mathcal{D}}$ and q_π, this sampler is $(q_{\mathcal{D}}, q_\pi)$-legitimate. This is because even if $\mathcal{A}$ knows the entire description of π, $\mathcal{S}$ always outputs an independent uniformly distributed n-bit string X conditioned on $\pi(X)$ having the first bit equal 0, and the distribution is uniform over 2^{n-1} possible such X's. Yet, given X sampled from $\mathcal{D}$ (and thus from $\mathcal{S}$), it is very easy to distinguish $\mathsf{H}^\pi(\mathsf{X})$ and $\mathsf{U}_{n/2}$ with advantage $\frac{1}{2}$, by having $\mathcal{A}$ simply output the first bit of its input, and thus *without even making a query to* π*!*

We stress that this is nothing more than the ideal-model analogue of the classical textbook proof that seedless extractors cannot exist for the class all k-sources, even when k is as large as $n - 1$. Above all, this shows that our class of legitimate samplers is strong enough to encompass such pathological examples, thus allowing to eliminate the odd artificiality of ideal models.

A brief discussion. The example above shows that our notion is strong enough to include (1) non-trivial distributions forcing us to use seeds and (2) permutation-independent samplers. It is meaningful to ask whether it is possible to weaken the requirement so that the output of $\mathcal{S}_{i^*}$ is only unpredictable when the π queries issued by $\mathcal{S}_j$ for $j \neq i^*$ and by $\mathcal{D}$ are not revealed by the game, and still get meaningful results. We believe this is possible in general, but without restrictions, there are non-trivial dependencies arising (thanks to the auxiliary input) between what the adversary can see and the sampling of I_{i^*} which we cannot handle in our proofs in a *generic* way.

3.2 Robustness, Recovering and Preserving Security in the IPM

Robustness. The definition of robustness follows the one from [15], with the aforementioned modifications tailored at our setting.

The formal definition of robustness is based on the game ROB given in Figure 2 and parametrized by a constant γ^*. The game description consists of special procedures initialize and finalize and 5 additional oracles. It is run as follows: first the initialize procedure is run, its output is given to the adversary which is then

Procedure initialize:
$\pi \xleftarrow{\$} \mathsf{Perms}(n)$
$\mathsf{seed} \xleftarrow{\$} \mathsf{setup}^{\pi}()$
$S \xleftarrow{\$} 0^r \,\|\, \{0,1\}^c$
$\sigma \leftarrow \perp$
$\mathsf{corrupt} \leftarrow \mathtt{false}$
$e \leftarrow c$
$b \xleftarrow{\$} \{0,1\}$
return seed

Procedure finalize(b^*):
return $(b = b^*)$

Procedure $\mathcal{D}$-refresh:
$(\sigma, I, \gamma, z) \xleftarrow{\$} \mathcal{D}^{\pi}(\sigma)$
$S \leftarrow \mathsf{refresh}^{\pi}(\mathsf{seed}, S, I)$
$e \leftarrow e + \gamma$
if $e \geq \gamma^*$:
  $\mathsf{corrupt} \leftarrow \mathtt{false}$
return (γ, z)

Procedure next-ror:
$(S, R_0) \xleftarrow{\$} \mathsf{next}^{\pi}(\mathsf{seed}, S)$
$R_1 \xleftarrow{\$} \{0,1\}^{\ell}$
if $\mathsf{corrupt} = \mathtt{true}$:
  $e \leftarrow 0$
  return R_0
return R_b

Procedure get-next:
$(S, R) \xleftarrow{\$} \mathsf{next}^{\pi}(\mathsf{seed}, S)$
if $\mathsf{corrupt} = \mathtt{true}$:
  $e \leftarrow 0$
return R

Procedure get-state:
$e \leftarrow 0$
$\mathsf{corrupt} \leftarrow \mathtt{true}$
return S

Procedure set-state(S^*):
$e \leftarrow 0$
$\mathsf{corrupt} \leftarrow \mathtt{true}$
$S \leftarrow S^*$

Fig. 2. Definition of the game $\mathsf{ROB}^{\gamma^*}_{\mathbf{G}}(\mathcal{A}, \mathcal{D})$.

allowed to query the 5 oracles described, in addition to π and π^{-1}, and once it outputs a bit b^*, this is then given to the finalize procedure, which generates the final output of the game.

For an adversary $\mathcal{A}$ and a distribution sampler $\mathcal{D}$, the advantage against the robustness of a PRNG with input $\mathbf{G}$ is defined as

$$\mathsf{Adv}^{\gamma^*\text{-rob}}_{\mathbf{G}}(\mathcal{A}, \mathcal{D}) = 2 \cdot \Pr\left[\mathsf{ROB}^{\gamma^*}_{\mathbf{G}}(\mathcal{A}, \mathcal{D}) \Rightarrow 1\right] - 1 .$$

An adversary against robustness that asks q_π queries to its π/π^{-1} oracles, $q_{\mathcal{D}}$ queries to its $\mathcal{D}$-refresh oracle, q_R queries to its next-ror/get-next oracles, and q_S queries to its get-state/set-state oracles, is called a $(q_\pi, q_{\mathcal{D}}, q_R, q_S)$-adversary.

Recovering security. We follow the definition from [15], again adapted to our setting. In particular, we only require that the state resulting from the final next call in the experiment has to be indistinguishable from a c-bit uniformly random string preceded by r zeroes, instead of a random n-bit string.

Recovering security is defined in terms of the game REC parametrized by $q_{\mathcal{D}}$, γ^*, given in Figure 3. For an adversary $\mathcal{A}$ and a distribution sampler $\mathcal{D}$, the advantage against the recovering security of a PRNG with input $\mathbf{G}$ is defined as

$$\mathsf{Adv}^{(\gamma^*, q_{\mathcal{D}})\text{-rec}}_{\mathbf{G}}(\mathcal{A}, \mathcal{D}) = 2 \cdot \Pr\left[\mathsf{REC}^{\gamma^*, q_{\mathcal{D}}}_{\mathbf{G}}(\mathcal{A}, \mathcal{D}) \Rightarrow 1\right] - 1 .$$

An adversary against recovering security that asks q_π queries to its π/π^{-1} oracles is called a q_π-adversary.

Preserving security. We again follow the definition from [15], with similar modifications as in the case of recovering security above.

The formal definition of preserving security is based on the game PRES given in Figure 4. For an adversary $\mathcal{A}$, the advantage against the preserving security

1. The challenger chooses $\pi \xleftarrow{\$} \mathsf{Perms}(n)$, $\mathsf{seed} \xleftarrow{\$} \mathsf{setup}^{\pi}()$, and $b \xleftarrow{\$} \{0,1\}$ and sets $\sigma_0 \leftarrow \perp$. For $k = 1, \ldots, q_{\mathcal{D}}$, the challenger computes $(\sigma_k, I_k, \gamma_k, z_k) \leftarrow \mathcal{D}^{\pi}(\sigma_{k-1})$.
2. The attacker $\mathcal{A}$ gets seed and $\gamma_1, \ldots, \gamma_{q_{\mathcal{D}}}, z_1, \ldots, z_{q_{\mathcal{D}}}$. It also gets access to oracles π/π^{-1}, and to an oracle $\mathsf{get\text{-}refresh}()$ which initially sets $k \leftarrow 0$ and on each invocation increments $k \leftarrow k+1$ and outputs I_k. At some point, $\mathcal{A}$ outputs a value $S_0 \in \{0,1\}^n$ and an integer d such that $k + d \leq q_{\mathcal{D}}$ and $\sum_{j=k+1}^{k+d} \gamma_j \geq \gamma^*$.
3. For $j = 1, \ldots, d$ the challenger computes $S_j \leftarrow \mathsf{refresh}^{\pi}(\mathsf{seed}, S_{j-1}, I_{k+j})$. If $b = 0$ it sets $(S^*, R) \leftarrow \mathsf{next}^{\pi}(\mathsf{seed}, S_d)$, otherwise it sets $(S^*, R) \xleftarrow{\$} (0^r \parallel \{0,1\}^c) \times \{0,1\}^r$. The challenger gives $I_{k+d+1}, \ldots, I_{q_{\mathcal{D}}}$ and (S^*, R) to $\mathcal{A}$.
4. The attacker again gets access to π/π^{-1} and outputs a bit b^*. The output of the game is 1 iff $b = b^*$.

Fig. 3. Definition of the game $\mathsf{REC}_{\mathbf{G}}^{\gamma^*, q_{\mathcal{D}}}$.

1. The challenger chooses $\pi \xleftarrow{\$} \mathsf{Perms}(n)$, $\mathsf{seed} \xleftarrow{\$} \mathsf{setup}^{\pi}()$ and $b \xleftarrow{\$} \{0,1\}$ and a state $S_0 \xleftarrow{\$} 0^r \parallel \{0,1\}^c$.
2. The attacker $\mathcal{A}$ gets access to oracles π/π^{-1}, and outputs a sequence of values $I_1, \ldots, I_d$ with $I_j \in \{0,1\}^*$ for all $j \in [d]$.
3. The challenger computes $S_j \leftarrow \mathsf{refresh}^{\pi}(\mathsf{seed}, S_{j-1}, I_j)$ for all $j = 1, \ldots, d$. If $b = 0$ it sets $(S^*, R) \leftarrow \mathsf{next}^{\pi}(\mathsf{seed}, S_d)$, otherwise it sets $(S^*, R) \xleftarrow{\$} (0^r \parallel \{0,1\}^c) \times \{0,1\}^r$. The challenger gives (S^*, R) to $\mathcal{A}$.
4. The attacker $\mathcal{A}$ again gets access to π/π^{-1} and outputs a bit b^*. The output of the game is 1 iff $b = b^*$.

Fig. 4. Definition of the game $\mathsf{PRES}_{\mathbf{G}}$.

of a PRNG with input $\mathbf{G}$ is defined as

$$\mathsf{Adv}_{\mathbf{G}}^{\mathsf{pres}}(\mathcal{A}) = 2 \cdot \Pr\left[\mathsf{PRES}_{\mathbf{G}}(\mathcal{A}) \Rightarrow 1\right] - 1 \,.$$

An adversary against preserving security that asks q_π queries to its π/π^{-1} oracles is again called a q_π-adversary.

Relationship to [29]. Our need to adapt the notions of [15] confirms that, as observed in [29], assuming that the internal state of a PRNG is pseudorandom is overly restrictive. Indeed, our formalization is a special case of the approach from [29] into the setting of sponge-based constructions, where the so-called *masking function* would be defined as sampling a random $S \in 0^r \parallel \{0,1\}^c$ (and preserving the counter j). Our notions would then correspond to the "boot-strapped" notions from [29] and moreover, our results on recovering security below indicate that a naturally-defined procedure setup (for generating the initial state as in [29]) would make this masking function satisfy the *honest-initialization* property.

COMBINING PRESERVING AND RECOVERING SECURITY. This theorem establishes the very useful property that, roughly speaking, the preserving security and the recovering security of a PRNG together imply its robustness. We postpone its proof (following [15]) to the full version.

Theorem 4. *Let $\mathbf{G}[\pi]$ be a PRNG with input that issues q_π^{ref} (resp. q_π^{nxt}) π-queries in each invocation of* refresh *(resp.* next*); and let $\overline{q}_\pi = q_\pi + Q(q_\mathcal{D})$. For every $(q_\pi, q_\mathcal{D}, q_R, q_S)$-adversary $\mathcal{A}$ against robustness and for every Q-distribution sampler $\mathcal{D}$, there exists a family of $(q_\pi + q_R \cdot q_\pi^{\mathsf{nxt}} + q_\mathcal{D} \cdot q_\pi^{\mathsf{ref}})$-adversaries $\mathcal{A}_1^{(i)}$ against recovering security and a family of $(\overline{q}_\pi + q_R \cdot q_\pi^{\mathsf{nxt}} + q_\mathcal{D} \cdot q_\pi^{\mathsf{ref}})$-adversaries $\mathcal{A}_2^{(i)}$ against preserving security (for $i \in \{1, \ldots, q_R\}$) such that*

$$\mathsf{Adv}_{\mathbf{G}}^{\gamma^*\text{-}\mathsf{rob}}(\mathcal{A}, \mathcal{D}) \leq \sum_{i=1}^{q_R} \left(\mathsf{Adv}_{\mathbf{G}}^{(\gamma^*, q_\mathcal{D})\text{-}\mathsf{rec}}(\mathcal{A}_1^{(i)}, \mathcal{D}) + \mathsf{Adv}_{\mathbf{G}}^{\mathsf{pres}}(\mathcal{A}_2^{(i)}) \right) .$$

4 Robust Sponge-based PRNG

We consider the following PRNG construction, using a permutation $\pi \in \mathsf{Perms}(n)$, and depending on parameters s and t. This construction is a seeded variant of the general paradigm introduced by Bertoni *et al.* [10], including countermeasures to prevent attacks against forward secrecy. As we will see in the proof, the parameters s and t are going to enforce increasing degrees of security.

THE CONSTRUCTION. Let $s, t \geq 1$, and $r \leq n$, let $c := n - r$. We define $\mathbf{SPRG}_{s,t,n,r} = (\mathsf{setup}, \mathsf{refresh}, \mathsf{next})$, where the three algorithms setup, refresh, next make calls to some permutation $\pi \in \mathsf{Perms}(n)$ and operate as follows:

Proc. $\mathsf{setup}^\pi()$:
- **for** $i = 0, \ldots, s-1$ **do**
  - $\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^r$
- $\mathsf{seed} \leftarrow (\mathsf{seed}_0, \ldots, \mathsf{seed}_{s-1})$
- $j \leftarrow 1$
- **return** seed

Proc. $\mathsf{refresh}^\pi(\mathsf{seed}, S, I)$:
- $I_1, \ldots, I_\ell \xleftarrow{r} I$
- $S_0 \leftarrow S$
- **for** $i = 1, \ldots, \ell$ **do**
  - $S_i \leftarrow \pi(S_{i-1} \oplus (I_i \oplus \mathsf{seed}_j \,\|\, 0^c))$
  - $j \leftarrow j + 1 \mod s$
- **return** S_ℓ

Proc. $\mathsf{next}^\pi(\mathsf{seed}, S)$:
- $S_0 \leftarrow \pi(S)$
- $R \leftarrow S_0[1 \ldots r]$
- **for** $i = 1, \ldots, t$ **do**
  - $S_i \leftarrow \pi(S_{i-1})$
  - $S_i[1 \ldots r] \leftarrow 0^r$
- $j \leftarrow 1$
- **return** (S_t, R)

Note that apart from the entropy pool S, the PRNG also keeps a counter j internally as a part of its state. This counter increases (modulo s) as blocks are processed via refresh, and gets resetted whenever next is called. We will often just write **SPRG**, omitting the parameters s, t, n, r whenever the latter are clear from the context. In particular, the parameter s determines the length of the seed in terms of r-bit blocks. The construction **SPRG** is depicted in Figure 5.

We also note that it is not hard to modify our treatment to allow for next outputting multiple r-bit blocks at once, instead of just one, and this length could be variable. This could be done by providing an additional input, indicating the number of desired blocks and this would ensure better efficiency. The bounds presented here would only be marginally affected by this, but we decided to keep the presentation simple in this paper.

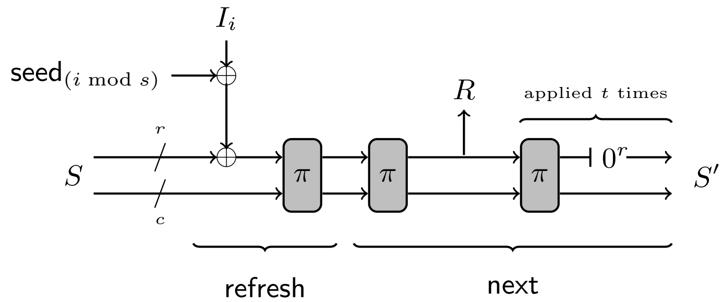

Fig. 5. Procedures refresh (processing a one-block input I_i) and next of the construction $\mathbf{SPRG}_{s,t}[\pi]$.

<u>INSECURITY OF THE UNSEEDED VERSION.</u> We show that seeding is necessary to achieve robustness. A similar argument implies that the original construction of [10] cannot be secure if the distribution sampler is allowed to depend on the public random permutation π.

To this end, we consider the distribution sampler $\mathcal{D}$ which on its first call outputs an $\ell \cdot r$-bit string, for a parameter ℓ such that $(\ell - 1)r \geq \gamma^*$. In particular, on its first call $\mathcal{D}$ simply outputs a source $\mathcal{S}_1$ which behaves as follows, given the corresponding entropy estimate $(\ell - 1) \cdot r$:

- It internally samples r-bit strings $I_1, \ldots, I_{\ell-1}$ uniformly at random.
- Then, it samples random $I_\ell^1, I_\ell^2, \ldots$ until it finds one such that $R^j[1] = 0$, where R are the r-bit returned by running next after running refresh, from the some initial state S, with inputs $I_1, \ldots, I_{\ell-1}, I_\ell^j$.

Additionally, consider a robustness adversary $\mathcal{A}$ that first calls set-state(S) and then $\mathcal{D}$-refresh(). Finally, it queries next-ror() obtaining R^*, and checks whether $R^*[1] = 0$. Clearly, $\mathcal{A}$ achieves advantage 1/2 despite $\mathcal{D}$ being legitimate.

5 Security Analysis of SPRG

This section gives a complete security analysis of **SPRG** given in Section 4 above, under the assumption that the underlying permutation is a random permutation $\pi \in \mathsf{Perms}(n)$. In particular, we prove the following theorem.

Theorem 5 (Security of SPRG). *Let* $\mathbf{SPRG} = \mathbf{SPRG}_{s,t,n,r}[\pi]$ *denote the PRNG given in Section 4. Let* $\gamma^* > 0$*, let* $\mathcal{A}$ *be a* $(q_\pi, q_\mathcal{D}, q_R, q_S)$*-adversary against robustness and let* $\mathcal{D}$ *be a* $(q_\mathcal{D}, q_\pi)$*-legitimate* Q*-distribution sampler such that the length of its outputs* $I_1, \ldots, I_{q_\mathcal{D}}$ *padded into* r*-bit blocks is at most* $\ell \cdot r$

bits in total. Then we have

$$\mathsf{Adv}^{\gamma^*\text{-rob}}_{\mathbf{SPRG}}(\mathcal{A},\mathcal{D}) \leq q_R \cdot \left(\frac{2(2\ell+2)(\overline{q}_\pi + q' + t + \ell) + 4\ell^2}{2^n} + \frac{\overline{q}_\pi + q' + t + 1}{2^{\gamma^*}} + \right. \\ \left. + \frac{22(\overline{q}_\pi + q' + t + 1)^2 + \overline{q}_\pi + q'}{2^c} + \frac{2(\overline{q}_\pi + q')}{2^{(r-1)t}} + \frac{Q(q_{\mathcal{D}})}{2^{sr}} \right) ,$$

where we use the notational shorthands $\overline{q}_\pi = q_\pi + Q(q_{\mathcal{D}})$ *and* $q' = (t+1)q_R + \ell$*.*

Note in particular that the construction is secure as long as $q_R \cdot \overline{q}_\pi \cdot \ell < 2^n$, $q_R \cdot \overline{q}_\pi, q_R^2 < 2^c$, $\overline{q}_\pi, q_R^2 \leq 2^{\gamma^*}$, $q_R^2, \overline{q}_\pi q_R \leq 2^{(r-1)t}$. Note that these are more than sufficient margins for SHA-3-like parameters, where $n = 1600$ and $c \geq 1024$ always holds. However, one should assess the bound more carefully for a single-key cipher instantiation, where $n = 128$. In this case, choosing a very small r (note that our construction and bound would support $r \geq 2$) would significantly increase the security margins.

The theorem follows from the bounds on recovering security and preserving security of **SPRG** proven in Lemmas 11 and 12 below, combined using Theorem 4. To establish these two bounds, we first give two underlying lemmas that represent the technical core of our analysis. The first one, Lemma 6, assesses the ability of a seeded sponge construction to act as a randomness extractor on inputs that are coming from a permutation-dependent distribution sampler. The second statement, given in Lemma 10, shows that the procedure next, given a high min-entropy input, produces an output that is very close to random.

5.1 The Sponge Extraction Lemma

The first part of our analysis addresses how the sponge structure can be used to extract (or in fact, condense) randomness. To this end, we first give a general definition of adaptively secure extraction functions.

Let $\mathbf{Ex}[\pi] : \{0,1\}^u \times \{0,1\}^v \times \{0,1\}^* \to \{0,1\}^n$ be an efficiently computable function taking as parameters a u-bit seed seed, a v-bit initialization value IV, together with an input string $X \in \{0,1\}^*$. It makes queries to a permutation $\pi \in \mathsf{Perms}(n)$ to produce the final n-bit output $\mathbf{Ex}^\pi(\mathsf{seed}, \mathsf{IV}, X)$. Then, for every $\gamma^* > 0$ and $q_{\mathcal{D}}$, for such an $\mathbf{Ex}$, an adversary $\mathcal{A}$ and a distribution sampler $\mathcal{D}$, we consider the game $\mathsf{GEXT}^{\gamma^*, q_{\mathcal{D}}}_{\mathbf{Ex}}(\mathcal{A},\mathcal{D})$ given in Figure 6. It captures the security of $\mathbf{Ex}$ in producing a random looking output in a setting where an *adaptive* adversary $\mathcal{A}$ can obtain side information and entropy estimates from a sampler $\mathcal{D}$, together with samples $I_1, \ldots, I_k$, until it commits on running $\mathbf{Ex}$ on adaptively chosen IV, as well as $I_{k+1} \ldots I_{k+d}$ for some d such that the guaranteed entropy of these values is $\sum_{i=k+1}^{k+d} \gamma_i \geq \gamma^*$. We define the $(q_{\mathcal{D}}, \gamma^*)$-*extraction advantage* of $\mathcal{A}$ and $\mathcal{D}$ against $\mathbf{Ex}$ as

$$\mathsf{Adv}^{(\gamma^*, q_{\mathcal{D}})\text{-ext}}_{\mathbf{Ex}}(\mathcal{A},\mathcal{D}) = 2 \cdot \Pr\left[\mathsf{GEXT}^{\gamma^*, q_{\mathcal{D}}}_{\mathbf{Ex}}(\mathcal{A},\mathcal{D}) \Rightarrow 1\right] - 1 .$$

Also, we denote by $\mathsf{Adv}^{(\gamma^*, q_{\mathcal{D}})\text{-hit}}_{n}(\mathcal{A},\mathcal{D})$ the probability that $\mathcal{A}$ queries $\pi^{-1}(Y^*)$ conditioned on $b = 1$ in game $\mathsf{GEXT}^{\gamma^*, q_{\mathcal{D}}}_{\mathbf{Ex}}(\mathcal{A},\mathcal{D})$ above, i.e., Y^* is the random

Game $\mathsf{GEXT}_{\mathbf{Ex}}^{\gamma^*,q_{\mathcal{D}}}(\mathcal{A},\mathcal{D})$:

1. The challenger chooses $\mathsf{seed} \xleftarrow{\$} \{0,1\}^u$, $\pi \xleftarrow{\$} \mathsf{Perms}(n)$ and $b \xleftarrow{\$} \{0,1\}$ and sets $\sigma_0 \leftarrow \perp$. For $k = 1, \ldots, q_{\mathcal{D}}$, the challenger computes $(\sigma_k, I_k, \gamma_k, z_k) \leftarrow \mathcal{D}^{\pi}(\sigma_{k-1})$.
2. The attacker $\mathcal{A}$ gets seed and $\gamma_1, \ldots, \gamma_{q_{\mathcal{D}}}, z_1, \ldots, z_{q_{\mathcal{D}}}$. It gets access to oracles π/π^{-1}, and to an oracle $\mathsf{get\text{-}refresh}()$ which initially sets $k \leftarrow 0$ and on each invocation increments $k \leftarrow k+1$ and outputs I_k. At some point, $\mathcal{A}$ outputs a value IV and an integer d such that $k + d \leq q_{\mathcal{D}}$ and $\sum_{j=k+1}^{k+d} \gamma_j \geq \gamma^*$.
3. If $b = 1$, we set $Y^* \xleftarrow{\$} \{0,1\}^n$, and if $b = 0$, we let $Y^* \leftarrow \mathbf{Ex}^{\pi}(\mathsf{seed}, \mathsf{IV}, I_{k+1} \parallel \cdots \parallel I_{k+d})$. Then, the challenger gives back Y^* and $I_{k+d+1}, \ldots, I_{q_{\mathcal{D}}}$ to $\mathcal{A}$.
4. The attacker again gets access to π/π^{-1} and outputs a bit b^*. The output of the game is 1 iff $b = b^*$.

Fig. 6. Definition of the game $\mathsf{GEXT}_{\mathbf{Ex}}^{\gamma^*,q_{\mathcal{D}}}(\mathcal{A},\mathcal{D})$.

n-bit challenge. (The quantity really only depends on n, and not on the actual function $\mathbf{Ex}$, which is dropped from the notation.) Note that in general $\mathsf{Adv}_n^{(\gamma^*,q_{\mathcal{D}})-\mathsf{hit}}(\mathcal{A},\mathcal{D})$ can be one, but we will only consider it for specific adversaries $\mathcal{A}$ for which it can be argued to be small, as we discuss below.

SPONGE-BASED EXTRACTION. We consider a sponge-based instantiation of $\mathbf{Ex}$. That is, for parameters $r \leq n$ (recall that we use the shorthand $c = n - r$), we consider the construction $\mathbf{Sp}_{n,r,s}[\pi] : \{0,1\}^{s \cdot r} \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ using a permutation $\pi \in \mathsf{Perms}(n)$ which, given seed $\mathsf{seed} = (\mathsf{seed}_0, \ldots, \mathsf{seed}_{s-1})$ (where $\mathsf{seed}_i \in \{0,1\}^r$ for all i), initialization value $\mathsf{IV} \in \{0,1\}^n$, input $X \in \{0,1\}^*$, first encodes X into r-bit blocks $X_1, \ldots, X_\ell$, and then outputs Y_ℓ, where $Y_0 \leftarrow \mathsf{IV}$ and for all $i \in [\ell]$,

$$Y_i \leftarrow \pi(Y_{i-1} \oplus (X_i \oplus \mathsf{seed}_{i \bmod s}) \parallel 0^c) .$$

We now turn to the following lemma, which establishes that the above construction $\mathbf{Sp}_{n,r,s}[\pi]$ is indeed a good extractor with respect to the notion defined above, as long as $\mathsf{Adv}_n^{(\gamma^*,q_{\mathcal{D}})-\mathsf{hit}}(\mathcal{A},\mathcal{D})$ is sufficiently small – a condition that will hold in applications of this lemma.

Lemma 6 (Extraction Lemma). *Let r, s be integers, let $q_{\mathcal{D}}, q_\pi$ be arbitrary, and let $\gamma^* > 0$. Also, let $\mathcal{D}$ be a $(q_{\mathcal{D}}, q_\pi)$-legitimate Q-distribution sampler, such that the length of its outputs $I_1, \ldots, I_{q_{\mathcal{D}}}$ padded into r-bit blocks is at most $\ell \cdot r$*

bits in total. Then, for any adversary $\mathcal{A}$ making $q_\pi \leq 2^{c-2}$ queries,

$$\mathsf{Adv}^{(\gamma^*,q_{\mathcal{D}})-\mathsf{ext}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A},\mathcal{D}) \leq \frac{\overline{q}_\pi}{2^{\gamma^*}} + \frac{Q(q_{\mathcal{D}})}{2^{sr}} + \frac{14\overline{q}_\pi^2}{2^c} + \frac{2\overline{q}_\pi \ell + 2\ell^2}{2^n} + \mathsf{Adv}^{(\gamma^*,q_{\mathcal{D}})-\mathsf{hit}}_{n}(\mathcal{A},\mathcal{D}) \; , \quad (3)$$

where $\overline{q}_\pi = q_\pi + Q(q_{\mathcal{D}})$.

<u>Discussion.</u> Once again, we note that in (3), we cannot *in general* expect the advantage $\mathsf{Adv}^{(\gamma^*,q_{\mathcal{D}})-\mathsf{hit}}_{n}(\mathcal{A},\mathcal{D})$ to be small – any $\mathcal{A}$ *sees* Y^* and thus *can* query it, and the bound is hence void for such adversaries. The reason why this is not an issue in our context is that the extraction lemma will be applied to *specific* $\mathcal{A}$'s resulting from reductions in scenarios where $\mathbf{Sp}_{n,r,s}$ is used to derive a key for an algorithm which *is already secure* when used with a proper independent random key. In this case, it is usually easy to upper bound $\mathsf{Adv}^{(\gamma^*,q_{\mathcal{D}})-\mathsf{hit}}_{n}(\mathcal{A},\mathcal{D})$ in terms of the probability of a certain adversary $\mathcal{A}'$ (from which $\mathcal{A}$ is derived) recovering the secret key of a secure construction.

But why is this term necessary? We note that one *can* expect the output to be random even without this restriction on querying $\pi^{-1}(Y)$, if we have the guarantee that the weakly random input fed into $\mathbf{Sp}_{n,r,s}$ is long enough. However, this only yields a weaker result. In particular, if $\mathbf{Sp}_{n,r,s}$ is run on r-bit inputs $I_{k+1},\ldots,I_{k+d}$ to produce an output Y^* (which may be replaced by a random one in the case $b = 1$), it is not hard to see that guessing $I_{k+2},\ldots,I_{k+d}$ is sufficient to distinguish, regardless of I_{k+1}. This is because an adversary $\mathcal{A}$ can simply "invert" the construction starting from computing $S_{k+d-1} \leftarrow \pi^{-1}(Y^*)$, $S_{k+d-2} \leftarrow \pi^{-1}(S_{k+d-1} \oplus (I_{k+d} \oplus \mathsf{seed}_{k+d \bmod s}) \,\|\, 0^c)$, … until it recovers S_0, and then checks whether $S_0[r+1\ldots n] = \mathsf{IV}[r+1\ldots n]$. This will always succeed (given the right guess) in the $b=0$ case, but with small probability in the $b=1$ case. Above all, the crucial point is that I_{k+1} is not necessary to perform this attack. In particular, this would render the result useless for $d = 1$, whereas our statement still makes it useful as long as $q_\pi \leq 2^r$, which is realistic for say $r \geq 80$, and $\mathsf{Adv}^{(\gamma^*,q_{\mathcal{D}})-\mathsf{hit}}_{n}(\mathcal{A},\mathcal{D})$ is small.

An independent observation is that for oracle-independent distribution samplers (i.e., which do not make any permutation queries), we have $Q(q_{\mathcal{D}}) = 0$. In this case, the bound becomes independent of s, and indeed one can show that the bound holds even if the seed is contant (i.e., all zero), capturing the common wisdom that seeding is unnecessary for oracle-independent distributions.

<u>Proof intuition.</u> The proof of Lemma 6, which we give in full detail below, is inspired by previous analyses of keyed sponges, which can be seen as a special case where a truly random input is fed into $\mathbf{Sp}_{n,r,s}$.[7] We will show that the advantage of $\mathcal{A}$ and $\mathcal{D}$ is bounded roughly by the probability that they jointly succeed in making all queries necessary to compute $\mathbf{Sp}_{n,r,s}(\mathsf{seed},\mathsf{IV},I_{k+1}\,\|\,\ldots\,\|\,I_{k+d})$. Indeed, we show that as long as not all necessary queries are made, then the

[7] We note that none of these analysis tried to capture a general statement.

distinguisher cannot tell apart the case $b = 0$ from the case $b = 1$ with substantial advantage. The core of the proof is bounding the above probability that all queries are issued.

To this end, with $X_1, \ldots, X_\ell$ representing the encoding into r-bit blocks of $I_{k+1} \| \ldots \| I_{k+d}$, we consider all possible sequences of ℓ queries to the permutation, each made by $\mathcal{A}$ *or* $\mathcal{D}$, resulting in (not necessarily all distinct) input-output pairs $(\alpha_1, \beta_1), \ldots, (\alpha_\ell, \beta_\ell)$ with the property that

$$\alpha_i[r+1 \ldots n] = \beta_{i-1}[r+1 \ldots n]$$

for every $i \in [\ell]$, where we have set $\beta_0 = \mathsf{IV}$ for notational compactness. We call such sequence of ℓ input-output pairs a *potential chain*. We are interested in the probability that for *some* potential chain we additionally have

$$\alpha_i[1 \ldots r] = \beta_{i-1}[1 \ldots r] \oplus X_i \oplus \mathsf{seed}_{i \bmod s} \tag{4}$$

for all $i \in [\ell]$. Let us see why we can expect the probability that this happens to be small.

Recall that our structural restriction on $\mathcal{D}$ enforces that all of the values $I_{k+1}, \ldots, I_{k+d}$ are explicitly sampled by component sources $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_{k+d}$. One first convenient observation is that as long as the overall number of permutation queries by $\mathcal{D}$ and $\mathcal{A}$, which is denoted by $\bar{q}_\pi$, is smaller than roughly $2^{c/2}$, then every potential chain can have only one of the two following formats:

- *Type A chains.* For $k \in [0 \ldots \ell]$, k input-output pairs $(\alpha_1, \beta_1), \ldots, (\alpha_k, \beta_k)$ resulting from *forward* queries made by $\mathcal{D}$ *outside* the process of sampling $I_{k+1} \ldots I_{k+d}$ by $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_{k+d}$, followed by $\ell - k$ more input-output pairs $(\alpha_{k+1}, \beta_{k+1}), \ldots, (\alpha_\ell, \beta_\ell)$ resulting from queries made by $\mathcal{A}$ directly.
- *Type B chains.* The potential chain is made by some input-output pairs $(\alpha_1, \beta_1), \ldots, (\alpha_\ell, \beta_\ell)$ all resulting from *forward* permutation queries made by $\mathcal{D}$, in particular also possibly by the component sources $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_{k+d}$.

One can also show that for $\overline{q}_\pi < 2^{c/2}$, it is likely that the number of such potential chains (either of Type A or Type B) is at most $\overline{q}_\pi$ and $Q(q_\mathcal{D})$, respectively. Now, we can look at the process of creating Type A and Type B chains *separately*, and note that in the former, the outputs of $\mathcal{S}_{k+1}, \ldots, S_{k+d}$ have some uncertainty left (roughly, at least γ^* bits of entropy), thus the generated $X_1, \ldots, X_\ell$ end up satisfying (4) for each of the Type A potential chains with probability at most $2^{-\gamma^*}$. Symmetrically, the process of generating Type B chains is totally independent of the seed, and thus once the seed is chosen (which is made of $s \cdot r$ random bits), each one of the at most $Q(q_\mathcal{D})$ potential Type B chains ends up satisfying (4) with probability upper bounded by roughly 2^{-rs}.

We stress that making this high-level intuition formal is quite subtle.

<u>Can we achieve a better bound?</u> The extraction lemma requires $\overline{q}_\pi \leq 2^{c/2}$ for it to be meaningful. One can indeed hope to extend the techniques from [14] and obtain a result (at least for permutation-independent sources) which holds even if π is *completely known* to $\mathcal{A}$, while still being randomly sampled. However,

in this regime one can expect the state output by $\mathbf{Sp}_{n,r,s}$ to be random only as long as at least n random bits have been input. In contrast, here we aim at the heuristic expectation (formalized in the ideal model) that as long as the number of queries is small compared to the entropy of the distribution, then the output looks random.

The restriction $q_\pi \leq 2^{c/2}$ is common for sponges – beyond this, collisions become easy to find, and parameters are set to prevent this. Nonetheless, recent analyses of key absorption (which can be seen as a special case where the inputs are uniform) in sponge-based PRFs [20] trigger hope for security for nearly all $q_\pi \leq 2^c$, as they show that such collisions are by themselves not harmful. Unfortunately, in such high query regimes the number of potential chains as described above effectively explodes, and using the techniques of [20] (which are in turn inspired by [12]) to bound this number results in a fairly weak result.

Proof (of Lemma 6). The proof uses the H-coefficient method, as illustrated in Section 2 – indeed, to upper bound $\mathsf{Adv}^{(\gamma^*,q_\mathcal{D})-\mathsf{ext}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A},\mathcal{D})$, by a standard argument, one needs to upper bound the difference between the probabilities that $\mathcal{A}$ outputs 1 in the $b=1$ and in the $b=0$ cases, respectively. Throughout this proof, we assume that $\mathcal{A}$ is deterministic, and that $\mathcal{D}$ is also deterministic, up to being initialized with a random input R (of sufficient length) consisting of all random coins used by $\mathcal{D}$. In particular, R also contains the random coins used to sample the $I_1, I_2, \ldots, I_{q_\mathcal{D}}$ values by the sources $\mathcal{S}_1, \ldots, \mathcal{S}_{q_\mathcal{D}}$ output by $\mathcal{D}$.

To simplify the proof, we enhance the game $\mathsf{GEXT}^{\gamma^*,q_\mathcal{D}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A},\mathcal{D})$ so that the adversary $\mathcal{A}$, when done interacting with π, learns some extra information just before outputting the decision bit b'. This extra information includes:

- All strings $I_{k+1}, \ldots, I_{k+d}$ generated by $\mathcal{D}$ and hidden to $\mathcal{A}$ so far.
- The randomness R and all queries to π made by the distribution sampler $\mathcal{D}$ throughout its $q_\mathcal{D}$ calls. This includes all queries made by $\mathcal{S}_1, \ldots, \mathcal{S}_{q_\mathcal{D}}$. Recall that there are at most $Q(q_\mathcal{D})$ such queries by definition.

While this extra information is substantial, note that $\mathcal{A}$ cannot make any further queries to the random permutation after learning it, and, as we will see, this information does not hurt indistinguishability. Introducing it will make reasoning about the proof substantially easier. To start with, note that an execution of $\mathsf{GEXT}^{\gamma^*,q_\mathcal{D}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A},\mathcal{D})$ defines a *transcript* of the form

$$\tau = ((u_1,v_1),\ldots,(u_{q'},v_{q'}), Y^*, R, \mathsf{seed}=(\mathsf{seed}_1,\ldots,\mathsf{seed}_s), \\ \gamma_1,\ldots,\gamma_{q_\mathcal{D}}, I_1,\ldots,I_{q_\mathcal{D}}, z_1\ldots z_{q_\mathcal{D}}, \mathsf{IV}, k, d)\ , \quad (5)$$

where (u_i,v_i) are the input-output pairs resulting from the π-queries by $\mathcal{D}$ and $\mathcal{A}$ (that is, either $\pi(u_i)=v_i$ or $\pi^{-1}(v_i)=u_i$ for each (u_i,v_i) was queried by at least one of $\mathcal{D}$ and $\mathcal{A}$), removing duplicates, and ordered lexicographically. Note in particular that $q' \leq Q(q_\mathcal{D}) + q_\pi = \overline{q}_\pi$, and that the information whether a pair is the result of a forward or a backward query (or both) is omitted from the transcript, as it will not be used explicitly in the following.

We say that a transcript τ as in (5) is *valid* if when running $\mathsf{GEXT}^{\gamma^*,q_{\mathcal{D}}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A},\mathcal{D})$ with seed value fixed to seed, feeding Y^* to $\mathcal{A}$, executing $\mathcal{D}$ with randomness R, and answering permutation queries via a partial permutation π' such that $\pi'(u_i) = v_i$ for all $i \in [q']$, then

- The execution terminates, i.e., every permutation query is on a point for which π' is defined. Moreover, *all* queries in $(u_1, v_1), \ldots, (u_{q'}, v_{q'})$ are asked by either $\mathcal{D}$ or $\mathcal{A}$ at some point.
- $\mathcal{D}$ indeed outputs $(I_1, z_1, \gamma_1), \ldots, (I_{q_\mathcal{D}}, z_{q_\mathcal{D}}, \gamma_{q_\mathcal{D}})$.
- $\mathcal{A}$ indeed outputs IV and d, after k calls to $\mathsf{get\text{-}refresh}()$.

Now let T_0 and T_1 be the distributions on valid transcripts resulting from $\mathsf{GEXT}^{\gamma^*,q_{\mathcal{D}}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A},\mathcal{D})$ in the $b = 0$ and $b = 1$ cases, respectively. Then,

$$\mathsf{Adv}^{(\gamma^*,q_{\mathcal{D}})-\mathsf{ext}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A},\mathcal{D}) \leq \mathbf{SD}(\mathsf{T}_0, \mathsf{T}_1) \;, \tag{6}$$

since the extra information can only help, and a (possibly non-optimal) distinguisher for T_0 and T_1 can still mimic $\mathcal{A}$'s original decision (i.e., output bit), ignoring all additional information contained in the transcripts.

We are now ready to present our partitioning of transcripts into good and bad transcripts. Note first that a transcript explicitly tells us the blocks $I_{k+1}, \ldots, I_{k+d}$ processed by $\mathbf{Sp}_{n,r,s}$, and concretely let $X_1 \ldots X_\ell$ be the enconding into r-bit blocks of $I_{k+1} \parallel \cdots \parallel I_{k+d}$ when processed by $\mathbf{Sp}_{n,r,s}$. In particular we let $\ell = \ell(\tau)$ be the length here (in terms of r-bit blocks) of this encoding.

Definition 7 (Bad transcript). *We say that a transcript τ as in (5) is* bad *if one of the two following properties is satisfied:*

- Hit. *There exists an (u_i, v_i), for $i \in [q']$, with $v_i = Y$. Note that this may be the result of a forward query $\pi(u_i)$ or a backward query $\pi^{-1}(v_i)$, or both. Which one is the case does not matter here.*
- Chain. *There exist ℓ permutation queries*

$$(\alpha_1, \beta_1), \ldots, (\alpha_\ell, \beta_\ell) \in \{(u_1, v_1), \ldots, (u_{q'}, v_{q'})\}$$

(not necessarily distinct) that constitute a chain, i.e., such that

$$\begin{aligned} \alpha_i[1 \ldots r] &= \beta_{i-1}[1 \ldots r] \oplus X_i \oplus \mathsf{seed}_{i \bmod s} \\ \alpha_i[r+1 \ldots n] &= \beta_{i-1}[r+1 \ldots n] \end{aligned} \tag{7}$$

for every $i \in [\ell]$, where we have set $\beta_0 = \mathsf{IV}$ for notational compactness.

Also, we denote by $\mathcal{B}$ the set of all bad transcripts.

The proof is then concluded by combining the following two lemmas using Theorem 1 in Section 2. The proofs of these lemmas are postponed to the full version.

Lemma 8 (Ratio analysis). *For all good transcripts τ,*

$$\mathsf{Pr}\left[\mathsf{T}_0 = \tau\right] \geq \left(1 - \frac{2q'\ell + 2\ell^2}{2^n}\right) \cdot \mathsf{Pr}\left[\mathsf{T}_1 = \tau\right] \ .$$

Lemma 9 (Bad event analysis). *For $\mathcal{B}$ as defined above,*

$$\mathsf{Pr}\left[\mathsf{T}_1 \in \mathcal{B}\right] \leq \frac{Q(q_{\mathcal{D}}) + q_\pi}{2^{\gamma^*}} + \frac{Q(q_{\mathcal{D}})}{2^{rs}} + \frac{14(Q(q_{\mathcal{D}}) + q_\pi)^2}{2^c} + \frac{2Q(q_{\mathcal{D}}) + q_\pi}{2^n} + \mathsf{Adv}_n^{(\gamma^*, q_{\mathcal{D}})-\mathsf{hit}}(\mathcal{A}, \mathcal{D}) \ .$$

□

5.2 Analysis of next

We now turn our attention to the procedure next. We are going to prove that if the input state to next has sufficient min-entropy, then the resulting state and the output bits are indistinguishable from a random element from $0^r \parallel \{0,1\}^c$ and $\{0,1\}^r$, respectively. The proof of the following lemma is postponed to the full version of this paper. We give an overview below.

Lemma 10 (Security of next). *Let S be a random variable on the n-bit strings. Then, for any q_π-adversary $\mathcal{A}$ and all $t \geq 1$,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dist}}(\mathsf{next}_t^\pi(\mathsf{S}), (0^r \parallel \mathsf{U}_c, \mathsf{U}_r)) \leq \frac{q_\pi}{2^{\mathbf{H}_\infty(\mathsf{S})}} + \frac{q_\pi}{2^{(r-1)t}} + \frac{4(q_\pi + t)^2}{2^c} + \frac{1}{2^n} \ , \quad (8)$$

where U_r and U_c are uniformly and independently distributed over the r- and c-bit strings, respectively.

<u>Proof outline.</u> Intuitively, given a value (S_t, R) output by either $\mathsf{next}(\mathsf{S})$ or simply by sampling it uniformly as in $0^r \parallel \mathsf{U}_c, \mathsf{U}_r$, the naive attacker would proceed as follows. Starting from S_t, it would try to guess the t r-bit parts in the computation of next (call them $Z_1, \ldots, Z_t$) which have been zeroed out, and repeatedly apply π^{-1} to recover the state S_0 (in the real case) which was used to generate the R-part of the output. Our proof will confirm that this attack is somewhat optimal, but one needs to exercise some care. Indeed, the proof will consist of two steps, which need to be made in the right order:

(1) We first show that if the attacker cannot succeed in doing the above, then it cannot distinguish whether it is given, together with R, the *actual* S_t value output by next on input S, or a value S'_t which is sampled independently of the internal workings of next (while still being given the actual R).
(2) We then show that given S'_t is now sampled independently of $\mathsf{next}(\mathsf{S})$, then the adversary will not notice a substantial difference if the *real* R-part of the output of $\mathsf{next}(\mathsf{S})$ (which is still given to $\mathcal{A}$) is finally replaced by an independently random one.

While (2) is fairly straightforward, the core of the proof is in (1). Similar to the proof of the extraction lemma, we are going to think here in terms of the adversary attempting to build some potential "chains" of values, which are sequences of queries (α_i, β_i) for $i \in [t]$ where $\beta_{i-1}[r+1 \ldots n] = \alpha_i[r+1 \ldots n]$ for all $i \geq 2$, $\alpha_i[1 \ldots r] = 0^r$ for all $i \geq 2$, and $\beta_t[r+1 \ldots n] = S_t[r+1 \ldots n]$. The adversary's hope is that one of these chains is such that $\beta_i[1 \ldots r] = Z_i$ for all $i \in [t]$, and this would allow to distinguish.

It is not hard to show that as long as $q_\pi \leq 2^{c/2}$, there are at most q_π potential chains with high probability. However, it is harder to argue that the probability that one of these potential chains really matchs the Z_i values is small when the adversary is given the real S_t output by $\mathsf{next}(\mathsf{S})$. This is because the values $Z_1, \ldots, Z_t$ are already fixed during the execution, and arguing about their conditional distribution is difficult. Rather, our proof (using the H-coefficient technique) shows that it suffices to analyze the probability that the adversary builds such a valid chain in the ideal world, where the adversary is given an independent S'_t. This analysis becomes much easier, as the values $Z_1, \ldots, Z_t$ can be sampled lazily after the adversary is done with its permutation queries, and they are essentially *random* and independent of the potential chains they can match.

5.3 Recovering Security

We now use the insights obtained in the previous sections to establish the recovering security of our construction **SPRG**. To slightly simplify the notation, let $\varepsilon_{\text{ext}}(q_\pi, q_\mathcal{D})$ denote the first four terms on the right-hand side of the bound (3) in Lemma 6 as a function of q_π and $q_\mathcal{D}$; and let $\varepsilon_{\text{next}}(q_\pi)$ denote the right-hand side of the bound (8) in Lemma 10 as a function of q_π.

Lemma 11. *Let $\mathbf{SPRG}_{s,t,n,r}$ be the PRNG given in Section 4 and let $\varepsilon_{\text{ext}}(\cdot,\cdot)$ and $\varepsilon_{\text{next}}(\cdot)$ be defined as above. Let $\gamma^* > 0$ and $q_\mathcal{D} \geq 0$, let $\mathcal{A}$ be a q_π-adversary against recovering security and $\mathcal{D}$ be a $(q_\mathcal{D}, q_\pi)$-legitimate Q-distribution sampler $\mathcal{D}$ such that the length of its outputs $I_1, \ldots, I_{q_\mathcal{D}}$ padded into r-bit blocks is at most $\ell \cdot r$ bits in total. Then we have*

$$\mathsf{Adv}^{(\gamma^*, q_\mathcal{D})\text{-rec}}_{\mathbf{SPRG}[\pi]}(\mathcal{A}, \mathcal{D}) \leq \varepsilon_{\text{ext}}(q_\pi + t + 1, q_\mathcal{D}) + 2\varepsilon_{\text{next}}(\overline{q}_\pi) + \frac{q_\pi}{2^n} \ ,$$

where $\overline{q}_\pi = q_\pi + Q(q_\mathcal{D})$.

Proof. Intuitively, we argue that due to the extractor properties of $\mathbf{Sp}_{n,r,s}$ shown in Lemma 6, the state S_d in the experiment $\mathsf{REC}^{\gamma^*, q_\mathcal{D}}_{\mathbf{SPRG}}$ (after processing the inputs hidden from the adversary) will be close to random; and due to Lemma 10 the output of next invoked on this state will be close to random as well.

More formally, we start by showing that there exists a $(q_\pi+t+1)$-adversary $\mathcal{A}_1$ and a $\overline{q}_\pi$-adversary $\mathcal{A}_2$ such that

$$\mathsf{Adv}^{(\gamma^*, q_\mathcal{D})\text{-rec}}_{\mathbf{SPRG}[\pi]}(\mathcal{A}, \mathcal{D}) \leq \mathsf{Adv}^{(\gamma^*, q_\mathcal{D})-\text{ext}}_{\mathbf{Sp}_{n,r,s}, \mathcal{D}}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{dist}}_{\mathcal{A}_2}(\mathsf{next}^\pi(\mathsf{U}_n), (0^r \parallel \mathsf{U}_c, \mathsf{U}_r)) \ , \tag{9}$$

where U_ℓ always denotes an independent random ℓ-bit string. Afterwards, we apply Lemmas 6 and 10 to upper-bound the two advantages on the right-hand side of (9).

Let $\mathcal{A}$ be the adversary against recovering security from the statement. Consider an adversary $\mathcal{A}_1$ against extraction that works as follows: Upon receiving seed, $\gamma_1, \ldots, \gamma_{q_\mathcal{D}}$, $z_1, \ldots, z_{q_\mathcal{D}}$ from the challenger, it runs the adversary $\mathcal{A}$ and provides it with these same values. During its run, $\mathcal{A}$ issues queries to the oracles π/π^{-1} and $\mathsf{get\text{-}refresh}$, which are forwarded by $\mathcal{A}_1$ to the equally-named oracles available to it. At some point, $\mathcal{A}$ outputs a pair (S_0, d), $\mathcal{A}_1$ responds by setting $\mathsf{IV} \leftarrow S_0$ and outputting (IV, d) to the challenger. Upon receiving Y^* and $I_{k+d+1}, \ldots, I_{q_\mathcal{D}}$ from the challenger, $\mathcal{A}_1$ computes $(S^*, R^*) \leftarrow \mathsf{next}(Y^*)$ and feeds both (S^*, R^*) and $I_{k+d+1}, \ldots, I_{q_\mathcal{D}}$ to $\mathcal{A}$. Then it responds to the π-queries of $\mathcal{A}$ as before, and upon receiving the final bit b^* from $\mathcal{A}$, $\mathcal{A}_1$ outputs the same bit. It is easy to verify the query complexity of $\mathcal{A}_1$.

For analysis, note that if the bit chosen by the challenger is $b = 0$, for $\mathcal{A}$ this is a perfect simulation of the recovering game $\mathsf{REC}^{\gamma^*, q_\mathcal{D}}_{\mathbf{SPRG}}$ with the challenge bit being also set to 0. On the other hand, if the challenger sets $b = 1$, $\mathcal{A}$ is given $(S^*, R^*) \leftarrow \mathsf{next}(\mathsf{U}_n)$ for an independent random n-bit string U_n, while the game $\mathsf{REC}^{\gamma^*, q_\mathcal{D}}_{\mathbf{SPRG}}$ with challenge bit set to 1 would require randomly chosen $(S^*, R^*) \stackrel{\$}{\leftarrow} (0^r \,\|\, \{0,1\}^c) \times \{0,1\}^r$ instead. The latter term in the bound (9) accounts exactly for this discrepancy – to see this, just consider an adversary $\mathcal{A}_2$ that simulates both $\mathcal{A}_1$ and the game $\mathsf{GEXT}^{\gamma^*, q_\mathcal{D}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A}_1, \mathcal{D})$ with $b = 1$, and then uses the dist-challenge instead of the challenge for $\mathcal{A}$.

We conclude by upper bounding the advantages on the right-hand side of (9). First, Lemma 6 gives us

$$\mathsf{Adv}^{(\gamma^*, q_\mathcal{D})-\mathsf{ext}}_{\mathbf{Sp}_{n,r,s}, \mathcal{D}}(\mathcal{A}_1) \leq \varepsilon_{\mathrm{ext}}(q_\pi + t + 1, q_\mathcal{D}) + \mathsf{Adv}^{(\gamma^*, q_\mathcal{D})-\mathsf{hit}}_{\mathcal{D}, n}(\mathcal{A}_1) \; .$$

It hence remains to bound $\mathsf{Adv}^{(\gamma^*, q_\mathcal{D})-\mathsf{hit}}_{\mathcal{D}, n}(\mathcal{A}_1)$, which is the probability that $\mathcal{A}_1$ queries $\pi^{-1}(Y^*)$ in the ideal-case $b = 1$ in $\mathsf{GEXT}^{\gamma^*, q_\mathcal{D}}_{\mathbf{Sp}_{n,r,s}}(\mathcal{A}, \mathcal{D})$. Note that (apart from forwarding $\mathcal{A}$'s π-queries) the only π-queries that $\mathcal{A}_1$ asks "itself" are to evaluate the call $\mathsf{next}(Y^*)$, and these are only forward queries. Therefore, it suffices to bound the probability that $\mathcal{A}$ queries $\pi^{-1}(Y^*)$ and $\mathcal{A}_1$ forwards this query. Since the only information related to Y^* that $\mathcal{A}$ obtains during this experiment is $(S^*, R^*) \leftarrow \mathsf{next}(Y^*)$, if we replace these values by randomly sampled $(S^*, R^*) \stackrel{\$}{\leftarrow} (0^r \,\|\, \{0,1\}^c) \times \{0,1\}^r$, the value Y^* will be completely independent of $\mathcal{A}$'s view. Therefore, again there exists a $\overline{q}_\pi$-adversary $\mathcal{A}_3$ (actually, $\mathcal{A}_3 = \mathcal{A}_2$) such that

$$\mathsf{Adv}^{(\gamma^*, q_\mathcal{D})-\mathsf{hit}}_{\mathcal{D}, n}(\mathcal{A}_1) \leq \frac{q_\pi}{2^n} + \mathsf{Adv}^{\mathsf{dist}}_{\mathcal{A}_3}(\mathsf{next}^\pi(\mathsf{U}_n), (0^r \,\|\, \mathsf{U}_c, \mathsf{U}_r)) \; .$$

Finally, by Lemma 10 for both $i \in \{2,3\}$ we have

$$\mathsf{Adv}^{\mathsf{dist}}_{\mathcal{A}_i}(\mathsf{next}^{\pi}(\mathsf{U}_n), (0^r \parallel \mathsf{U}_c, \mathsf{U}_r)) \leq \varepsilon_{\mathrm{next}}(\overline{q}_\pi) \leq \frac{\overline{q}_\pi}{2^{\mathbf{H}_\infty(\mathsf{U}_n)}} + \frac{\overline{q}_\pi}{2^{(r-1)t}} + \frac{4(\overline{q}_\pi + t)^2}{2^c} + \frac{1}{2^n} = \frac{\overline{q}_\pi + 1}{2^n} + \frac{\overline{q}_\pi}{2^{(r-1)t}} + \frac{4(\overline{q}_\pi + t)^2}{2^c} ,$$

which concludes the proof. □

5.4 Preserving Security

Here we proceed to establish also the preserving security of **SPRG**.

Lemma 12. *Let* $\mathbf{SPRG}[\pi]$ *be the PRNG given in Section 4, and let* $\varepsilon_{\mathrm{next}}(\cdot)$ *be defined as above. For every* q_π*-adversary* $\mathcal{A}$ *against preserving security, we have*

$$\mathsf{Adv}^{\mathsf{pres}}_{\mathbf{SPRG}[\pi]}(\mathcal{A}) \leq \varepsilon_{\mathrm{next}}(q_\pi) + \frac{q_\pi}{2^c} + \frac{(2d'+1)(q_\pi + d')}{2^n} \leq \leq \frac{(2d'+2)(q_\pi + d')}{2^n} + \frac{q_\pi}{2^{(r-1)t}} + \frac{4(q_\pi + t)^2 + q_\pi}{2^c} ,$$

where d' *is the number of* r*-bit blocks resulting from parsing* $\mathcal{A}$*'s output* $I_1, \ldots, I_d$*.*

Proof. Intuitively, the proof again consists of two steps: showing that (1) since the initial state S_0 is random and hidden from the adversary, the state S_d will most likely look random to it as well; and (2) if S_d is random, we can again rely on Lemma 10 to argue about the pseudorandomness of the outputs of next.

More formally, consider a game PRES' which is defined exactly as the game PRES in Fig. 4, except that instead of computing the value S_d iteratively in Step 3, we sample it freshly at random as $S_d \xleftarrow{\$} \{0,1\}^n$. Moreover, imagine the permutation π as being lazy-sampled in both games.

Let $\mathcal{A}$ be an adversary participating in the game $\mathsf{PRES}_{\mathbf{SPRG}[\pi]}$. Let $\mathcal{QR}^{(1)}_\pi$ denote the set of query-response pairs that the adversary $\mathcal{A}$ asks to π via its oracles π/π^{-1} in its first stage (before submitting $I_1, \ldots, I_d$). More precisely, let $\mathcal{QR}^{(1)}_\pi$ denote the set of pairs $(u, v) \in \{0,1\}^n \times \{0,1\}^n$ such that $\mathcal{A}$ in its first stage either asked the query $\pi(u)$ and received the response v, or asked the query $\pi^{-1}(v)$ and received the response u. Moreover, let us denote by $I'_1, \ldots, I'_{d'}$ the r-bit blocks resulting from parsing the inputs $I_1, \ldots, I_d$ in sequence, using the parsing mechanism from the $\mathsf{refresh}$ procedure. Finally, recall that "$\rightarrow$" denotes the output of the adversary, as opposed to the game output.

We first argue that

$$\mathsf{Pr}\left[\mathsf{PRES}_{\mathbf{SPRG}[\pi]}(\mathcal{A}) \rightarrow 0 \,\middle|\, b = 0\right] - \mathsf{Pr}\left[\mathsf{PRES}'_{\mathbf{SPRG}[\pi]}(\mathcal{A}) \rightarrow 0 \,\middle|\, b = 0\right] \leq \frac{q_\pi}{2^c} + \frac{(2d'+1)(q_\pi + d')}{2^n} . \quad (10)$$

To see this, first note that the value S_0 is chosen independently at random from the set $0^r \parallel \{0,1\}^c$ and hidden from the adversary. Therefore, we have

$$\mathsf{Pr}\left[\exists(u,v) \in \mathcal{QR}_\pi^{(1)} \colon S_0 \oplus ((I_1' \oplus \mathsf{seed}_1) \parallel 0^c) = u\right] \leq \frac{\left|\mathcal{QR}_\pi^{(1)}\right|}{2^c} \leq \frac{q_\pi}{2^c} \ .$$

If this does not happen, the first invocation of π during the sequence of evaluations of $\mathsf{refresh}$ on $I_1, \ldots, I_d$ will be on a fresh value and hence its output (call it S_1') will be chosen uniformly at random from the $2^n - \left|\mathcal{QR}_\pi^{(1)}\right| - 1$ unused values. Hence, again the probability that the next π-invocation will be on an already defined value is at most $2(q_\pi+1)/2^n$. This same argument can be used iteratively up to the final state S_d: with probability at least $1-q_\pi/2^c-2d'(q_\pi+d')/2^n$ all of the π-invocations used during the sequence of $\mathsf{refresh}$-calls will happen on fresh values, and therefore S_d will be also chosen uniformly at random from the set of at least $2^n - q_\pi - d'$ values. This means that in this case, the statistical distance of S_d in the game $\mathsf{PRES}_{\mathbf{SPRG}[\pi]}$ from S_d in the game $\mathsf{PRES}'_{\mathbf{SPRG}[\pi]}$ where it is chosen at random will be at most $(q_\pi + d')/2^n$. Put together, this proves (10).

Now we observe that there exists a q_π-adversary $\mathcal{A}'$ such that

$$\mathsf{Pr}\left[\mathsf{PRES}'_{\mathbf{SPRG}[\pi]}(\mathcal{A}) \to 0 \,\middle|\, b=0\right] - \mathsf{Pr}\left[\mathsf{PRES}'_{\mathbf{SPRG}[\pi]}(\mathcal{A}) \to 0 \,\middle|\, b=1\right] \leq$$
$$\leq \mathsf{Adv}^{\mathsf{dist}}_{\mathcal{A}'}(\mathsf{next}_t^\pi(\mathsf{U}_n), (0^r \parallel \mathsf{U}_c, \mathsf{U}_r)) \leq \varepsilon_{\text{next}}(q_\pi) \quad (11)$$

where U_ℓ denotes a uniformly random ℓ-bit string. Namely, it suffices to consider $\mathcal{A}'$ that runs the adversary $\mathcal{A}$ and simulates the game PRES' for it (except for the π-queries; also note that $\mathcal{A}'$ does not need to compute the sequence of $\mathsf{refresh}$-calls), then replaces the challenge for $\mathcal{A}$ by its own challenge, and finally outputs the complement of the bit $\mathcal{A}$ outputs.

The proof is finally concluded by combining the bounds (10) and (11) and observing that if $b=1$, the games PRES and PRES' are identical. □

6 Key-derivation Functions from Sponges

This section applies the sponge extraction lemma (Lemma 6) to key-derivation functions (KDFs). We follow the formalization of Krawczyk [22]. While the fact that sponges can be used as KDFs is widely believed thanks to the existing indifferentiability analysis [9], our treatment allows for a stronger result for adversarial and oracle-dependent distributions.

KDFS AND THEIR SECURITY. A *key derivation function* is an algorithm KDF : $\{0,1\}^s \times \{0,1\}^* \times \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$, where the first input is the *seed*, the second is the *source material*, the third is the *context variable*, and the fourth is the *output length*. In particular, for all $\mathsf{seed} \in \{0,1\}^s$, $W, C \in \{0,1\}^*$ and $\mathsf{len} \in \mathbb{N}$, we have $|\mathsf{KDF}(\mathsf{seed}, W, C, \mathsf{len})| = \mathsf{len}$, and moreover $\mathsf{KDF}(\mathsf{seed}, W, C, \mathsf{len}')$ is a prefix of $\mathsf{KDF}(\mathsf{seed}, W, C, \mathsf{len})$ for all $\mathsf{len}' \leq \mathsf{len}$.

Game $\mathsf{GKDF}_{\mathsf{KDF}}^{\gamma^*,q_{\mathcal{D}}}(\mathcal{A},\mathcal{D})$:

1. The challenger chooses $\mathsf{seed} \xleftarrow{\$} \{0,1\}^u$, $\pi \xleftarrow{\$} \mathsf{Perms}(n)$ and $b \xleftarrow{\$} \{0,1\}$ and sets $\sigma_0 \leftarrow \perp$. For $k = 1, \ldots, q_{\mathcal{D}}$, the challenger computes $(\sigma_k, I_k, \gamma_k, z_k) \leftarrow \mathcal{D}^{\pi}(\sigma_{k-1})$.
2. The attacker $\mathcal{A}$ gets seed and $\gamma_1, \ldots, \gamma_{q_{\mathcal{D}}}, z_1, \ldots, z_{q_{\mathcal{D}}}$. It gets access to oracles π/π^{-1}, and to an oracle $\mathsf{get\text{-}refresh}()$ which initially sets $k \leftarrow 0$ and on each invocation increments $k \leftarrow k+1$ and outputs I_k. At some point, $\mathcal{A}$ outputs an integer d such that $k + d \le q_{\mathcal{D}}$ and $\sum_{j=k+1}^{k+d} \gamma_j \ge \gamma^*$.
3. If $b = 1$, we let $F = \mathbf{RO}(\cdot,\cdot)$, and if $b = 0$, $F = \mathsf{KDF}^{\pi}(\mathsf{seed}, I_{k+1} \,\|\, \ldots \,\|\, I_{k+d}, \cdot, \cdot)$. Then, the challenger gives back $I_{k+d+1}, \ldots, I_{q_{\mathcal{D}}}$ to $\mathcal{A}$.
4. The attacker gets access to π/π^{-1}, and in addition to F, and outputs a bit b^*. The output of the game is 1 iff $b = b^*$.

Fig. 7. Definition of the game $\mathsf{GKDF}_{\mathsf{KDF},\mathcal{D}}^{\gamma^*,q_{\mathcal{D}}}(\mathcal{A})$. Here, $\mathbf{RO}$ is an oracle which associates with each string x a potentially infinitely long string $\rho(x)$, and on input (x, len), it returns the first len bits of $\rho(x)$.

We consider KDF constructions making calls to an underlying permutation $\pi \in \mathsf{Perms}(n)$.[8] We define security of KDFs in terms of a security game $\mathsf{GKDF}_{\mathsf{KDF}}^{\gamma^*,q_{\mathcal{D}}}(\mathcal{A},\mathcal{D})$ which is slightly more general than the one used in [22], and described in Figure 7. In particular, similar to GEXT above, the game considers an incoming stream of $q_{\mathcal{D}}$ weakly random values, coming from a legitimate and oracle-dependent distribution sampler, and the attacker can choose a subset of these values with sufficient min-entropy *adaptively* to derive randomness from, as long as these values are guaranteed to have (jointly) min-entropy at least γ^*. The game then requires the attacker $\mathcal{A}$, given seed, to distinguish $\mathsf{KDF}(\mathsf{seed}, I_{k+1} \,\|\, \cdots \,\|\, I_{k+d}, \cdot, \cdot)$ from $\mathbf{RO}(\cdot,\cdot)$, where the latter returns for every $X \in \{0,1\}^*$ and $\mathsf{len} \in \mathbb{N}$, the first len bits of an infinitely long stream $\rho(X)$ of random bits associated with X.

Then, the kdf advantage of $\mathcal{A}$ is

$$\mathsf{Adv}_{\mathsf{KDF}}^{(\gamma^*,q_{\mathcal{D}})-\mathsf{kdf}}(\mathcal{A},\mathcal{D}) = 2 \cdot \Pr\left[\mathsf{GKDF}_{\mathsf{KDF}}^{\gamma^*,q_{\mathcal{D}}}(\mathcal{A},\mathcal{D}) \Rightarrow 1\right] - 1 \,.$$

SPONGE-BASED KDF. We present a sponge based KDF construction – denoted $\mathbf{SpKDF}_{n,r,s}$ – that can easily be implemented on top of SHA-3. It depends on three parameters n, r, s, and uses a seed of length $k = r \cdot s$ bits, represented as $\mathsf{seed} = (\mathsf{seed}_0, \ldots, \mathsf{seed}_{s-1})$. It uses a permutation π, and given $W, C \in \{0,1\}^*$, and $\mathsf{len} \in \mathbb{N}$, it operates as follows: It first splits W and C into r-bit blocks $W_1 \ldots W_d$ and $C_1 \ldots C_{d'}$,[9] and then computes, starting with $S_0 = \mathsf{IV}$, the states

[8] Once again, our treatment easily extends to other ideal models, but we dispense here with a generalization to keep our treatment sufficiently compact.

[9] As in the original sponge construction, we need to assume that C is always encoded so that every block $C_i \neq 0^r$.

$S_1, \ldots, S_d, S_{d+1}, \ldots, S_{d+d'}$, where

$$S_i \leftarrow \pi((W_i \oplus \mathsf{seed}_{i \bmod s}) \,\|\, 0^c \oplus S_{i-1}) \text{ for all } i \in [d]$$
$$S_i \leftarrow \pi((C_i \,\|\, 0^c) \oplus S_{i-1}) \text{ for all } i \in [d+1 \ldots d+d']$$

Then, for $t := \lceil \mathsf{len}/r \rceil$, if $t \geq 2$, it computes the values $S_{d+d'+1}, \ldots, S_{d+d'+t-1}$ as $S_i \leftarrow \pi(S_{i-1})$ for $i \in [d+d'+1 \ldots d+d'+t-1]$. Finally, $\mathbf{SpKDF}^{\pi}_{n,r,s}(\mathsf{seed}, W, C, \mathsf{len})$ outputs the first len bits of $S_{d+d'}[1 \ldots r] \,\|\, \cdots \,\|\, S_{d+d'+t-1}[1 \ldots r]$.

Security of sponge-based KDF. The proof of the following theorem (given in the full version) is an application of the sponge extraction lemma (Lemma 6), combined with existing analyses of the PRF security of keyed sponges with variable-output-length [24].

Theorem 13 (Security of SpKDF). *Let r, s be integers, let $q_{\mathcal{D}}, q_\pi$ be arbitrary, and let $\gamma^* > 0$. Also, let $\mathcal{D}$ be a $(q_{\mathcal{D}}, q_\pi)$-legitimate Q-distribution sampler $\mathcal{D}$ for which the overall output length (when invoked $q_{\mathcal{D}}$ times) is at most $\ell \cdot r$ bits after padding. Then, for all adversaries $\mathcal{A}$ making $q_\pi \leq 2^{c-2}$ queries to π, and q queries to F, where every query to the latter results in an input C encoded into at most ℓ' r-bit blocks, and in an output of at most len bits, we have*

$$\mathsf{Adv}^{(\gamma^*, q_{\mathcal{D}})\text{-}\mathsf{kdf}}_{\mathbf{SpKDF}_{n,r,s}}(\mathcal{A}, \mathcal{D}) \leq \frac{\tilde{q}_\pi}{2^{\gamma^*}} + \frac{Q(q_{\mathcal{D}})}{2^{sr}} + \frac{14\tilde{q}_\pi^2 + 6q^2\bar{\ell} + 3q\bar{\ell}\tilde{q}_\pi}{2^c} + \frac{2\tilde{q}_\pi\ell + 2\ell^2 + 6q^2\bar{\ell}^2 + \tilde{q}_\pi}{2^n},$$

where $\tilde{q}_\pi = (q_\pi + Q(q_{\mathcal{D}}))(1 + 2\lceil \frac{n}{r} \rceil)$ and $\bar{\ell} = \ell + \ell' + \lceil \mathsf{len}/r \rceil$.

Acknowledgments. We thank Mihir Bellare for insightful comments in an earlier stage of this project. Peter Gaži is supported by the European Research Council under an ERC Starting Grant (259668-PSPC). Stefano Tessaro was partially supported by NSF grants CNS-1423566, CNS-1528178, and the Glen and Susanne Culler Chair.

References

1. Information Technology - Security Techniques - Random Bit Generation. ISO/IEC 18031, 2011.
2. Martin R. Albrecht, Pooya Farshim, Kenneth G. Paterson, and Gaven J. Watson. On cipher-dependent related-key attacks in the ideal-cipher model. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 128–145. Springer, Heidelberg, February 2011.
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. Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2011.
5. Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05*, pages 203–212. ACM Press, November 2005.
6. Elaine B. Barker and John M. Kelsey. Sp 800-90a. recommendation for random number generation using deterministic random bit generators. Technical report, Gaithersburg, MD, United States, 2012.
7. 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.
8. L Bello. Dsa-1571-1 openssl–predictable random number generator. *Debian Security Advisory*, 2008.
9. 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.
10. 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.
11. Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, Heidelberg, May 2014.
12. Yuanxi Dai and John Steinberger. Tight security bounds for multiple encryption. Cryptology ePrint Archive, Report 2014/096, 2014. http://eprint.iacr.org/2014/096.
13. Anand Desai, Alejandro Hevia, and Yiqun Lisa Yin. A practice-oriented treatment of pseudorandom number generators. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 368–383. Springer, Heidelberg, April / May 2002.
14. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 494–510. Springer, Heidelberg, August 2004.
15. Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud, and Daniel Wichs. Security analysis of pseudo-random number generators with input: /dev/random is not robust. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 647–658. ACM Press, November 2013.
16. Yevgeniy Dodis, Thomas Ristenpart, and Salil P. Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 618–635. Springer, Heidelberg, March 2012.
17. Yevgeniy Dodis, Adi Shamir, Noah Stephens-Davidowitz, and Daniel Wichs. How to eat your entropy and have it too - optimal recovery strategies for compromised RNGs. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 37–54. Springer, Heidelberg, August 2014.

18. Leo Dorrendorf, Zvi Gutterman, and Benny Pinkas. Cryptanalysis of the windows random number generator. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07*, pages 476–485. ACM Press, October 2007.
19. D. Eastlake, J. Schiller, and S. Crocker. Randomness Requirements for Security. RFC 4086 (Best Current Practice), June 2005.
20. 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.
21. John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In Serge Vaudenay, editor, *FSE'98*, volume 1372 of *LNCS*, pages 168–188. Springer, Heidelberg, March 1998.
22. Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, Heidelberg, August 2010.
23. 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.
24. Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of full-state keyed and duplex sponge: Applications to authenticated encryption. Technical report, Cryptology ePrint Archive, Report 2015/541, 2015, http://eprint. iacr. org, 2015. To appear at ASIACRYPT'15.
25. 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.
26. Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. Reverse-engineering a cryptographic rfid tag. In *USENIX security symposium*, volume 28, 2008.
27. Jacques Patarin. The "coefficients H" technique (invited talk). In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, Heidelberg, August 2009.
28. 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.
29. Thomas Shrimpton and R. Seth Terashima. A provable-security analysis of Intel's secure key RNG. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 77–100. Springer, Heidelberg, April 2015.
30. Dan Shumow and Niels Ferguson. On the possibility of a back door in the NIST SP800-90 Dual Ec Prng. CRYPTO 2007 Rump Session, August 2007.
31. Anthony Van Herrewege and Ingrid Verbauwhede. Software only, extremely compact, keccak-based secure prng on arm cortex-m. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, pages 111:1–111:6, New York, NY, USA, 2014. ACM.