Seedless Fruit is the Sweetest: Random Number Generation, Revisited

Sandro Coretti*
New York University
corettis@nyu.edu

Yevgeniy Dodis†
New York University
dodis@cs.nyu.edu

Harish Karthikeyan‡
New York University
karthik@cs.nyu.edu

Stefano Tessaro§
University of Washington
tessaro@cs.washington.edu

February 21, 2019

Abstract

The need for high-quality randomness in cryptography makes random-number generation one of its most fundamental tasks.

A recent important line of work (initiated by Dodis et al., CCS '13) focuses on the notion of *robustness* for *pseudorandom number generators (PRNGs) with inputs*. These are primitives that use various sources to accumulate sufficient entropy into a state, from which pseudorandom bits are extracted. Robustness ensures that PRNGs remain secure even under state compromise and adversarial control of entropy sources. However, the achievability of robustness inherently depends on a *seed*, or, alternatively, on an ideal primitive (e.g., a random oracle), independent of the source of entropy. Both assumptions are problematic: seed generation requires randomness to start with, and it is arguable whether the seed or the ideal primitive can be kept independent of the source.

This paper resolves this dilemma by putting forward new notions of robustness which enable both (1) *seedless* PRNGs and (2) *primitive-dependent* adversarial sources of entropy. To bypass obvious impossibility results, we make a realistic compromise by requiring that the source produce sufficient entropy *even* given its evaluations of the underlying primitive. We also provide natural, practical, and provably secure constructions based on hash-function designs from compression functions, block ciphers, and permutations. Our constructions can be instantiated with minimal changes to industry-standard hash functions SHA-2 and SHA-3, or HMAC (as used for the key derivation function HKDF), and can be downgraded to *(online) seedless randomness extractors*, which are of independent interest.

On the way we consider both a *computational* variant of robustness, where attackers only make a bounded number of queries to the ideal primitive, as well as a new *information-theoretic* variant, which dispenses with this assumption to a certain extent, at the price of requiring a high rate of injected weak randomness (as it is, e.g., plausible on Intel's on-chip RNG). The latter notion enables applications such as everlasting security.

Finally, we show that the CBC extractor, used by Intel's on-chip RNG, is provably insecure in our model.

*Supported by NSF grants 1314568 and 1619158.

†Partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546.

‡Supported by NSF grant 1619158.

§Partially supported by NSF grants CNS-1553758 (CAREER), CNS-1719146, CNS-1528178, and IIS-1528041, and by a Sloan Research Fellowship.

Contents

1 Introduction

Good random number generation is essential for cryptography and beyond. In practice, this difficult task is solved by a primitive called *pseudorandom number generator with input (PRNG)*, whose aim is to quickly accumulate entropy from various physical sources in the environment (such as keyboard presses, timing of interrupts, etc.) into the state of the PRNG and then convert this high-entropy state into (pseudo) random bits. In particular, entropy accumulation should never stop since one may need to recover from occasional compromises of the PRNG state. PRNGs are ubiquitous and have extensive applications. For example, virtually all operating systems come equipped with a PRNG; e.g., `/dev/random` [47] for Linux, Yarrow [33] for MacOs/iOS/FreeBSD, and Fortuna [23] for Windows [22], where the latter two make use of standard cryptographic primitives as part of their design. Still, as we will argue below in a much broader context, even these widely used PRNGs lack adequate theoretical understanding and analysis, which are critical if such PRNGs or their future tweaks continue to be used ubiquitously.

The situation is not better in terms of standardization efforts, where existing PRNG standards [31, 34, 21, 5] are less mature than those for most other cryptographic primitives. For starters, there has not been any rigorous competition soliciting PNRG designs, and big parts of the existing standards concentrate on the difficult (ad-hoc and non-cryptographic) problem of entropy estimation rather than entropy accumulation and extraction. More importantly, standardized cryptographic PRNG constructions are rather ad-hoc, have no clear security definition/model, often have confusing syntax, and sometimes have been broken by subsequent analyses of the cryptographic community. The most widely known example is the DualEC PRNG, which appeared in the first version of the NIST SP 800-90A standard [5] in 2005 and remained there for years—despite early warnings by [41, 43] and allowing potential exploitation [12]—until Snowden's revelations finally led to its deprecation. Recent work [48] identified a lot of gaps and imprecision (sometimes leading to attacks or security concerns) in the existing analyses and deployment for the other 3 PRNGs from the NIST SP 800-90A standard. In a similar vein, [42] found several gaps and misconceptions in previous analyses and security justifications for the popular Intel Secure Key Hardware PRNG introduced in 2011.

One of the main goals of this work is to reverse this poor state of affairs and to design a rigorous, theoretically sound model of PRNGs. This model should be general enough to incorporate practical entropy sources available in the real world, as well as to formally prove security of "good," widely used PRNGs against realistic attackers.

1.1 Previous Theoretical Models for PRNGs: Seeds

In view of their practical importance, we are certainly not the first to formally study PRNGs through a theoretical lens. Indeed, several theoretical models and analyses of PRNGs have appeared in the literature [1, 18, 42, 20, 25, 27]. While differing in various details, these important works share two key principles:

(a) The PRNG should work even against *adversarial* entropy sources, as long as such sources eventually provide enough entropy (such sources are called "legitimate" [18]);

(b) assuming more structure beyond entropy is undesirable and brittle,[1] as this requires a rather detailed understanding of one's entropy sources.

However, while such extremely minimalist assumptions make these PRNG models applicable to a wide variety of entropy sources, they also come with a subtle, but very important caveat: the

[1] We do, however, later discuss an interesting approach suggested by [3].

randomness extraction module *cannot* be deterministic, as deterministic extraction from *general* entropy sources is impossible [15]. As a result, the PRNGs studied by these works are *seeded* (with the seed somehow chosen at initialization), but the entropy sources are assumed to be *independent of the seed*. This modeling is inherited from the underlying problem of randomness extraction, where *seeded extractors* [39] indeed overcome the impossibility of deterministic (or "seedless") extraction from general entropy sources.

While natural and sufficient for some applications of extractors, we argue that the need for a seed seems rather problematic in the deployment of PRNGs. First, if the reason for random number generation is the lack of access to high-quality random bits, then we may not have any way to generate the seed. More importantly, even if we *can* generate a uniformly random seed, it is crucial for the analysis that (potentially adversarial) entropy sources remain *independent* of the seed, for otherwise the extractor guarantees are lost. For example, if physical entropy sources inside the computer are used, these sources may be affected by the internal computations of the PRNG itself, and thus there may be correlations between the seed and the sources. Moreover, for many seeded PRNGs, the attacker could obtain information about the seed by either directly reading it from memory, or indirectly when the recently compromised or rebooted RNG is called on "low-entropy" inputs (so the output is no longer random and leaks information about the seed; this is called "premature next" attack by [20]).

This means that it is certainly an issue if the seed is just generated once and for all (perhaps using an expensive source of randomness) and hard-coded within implementations to be used for all future randomness extractions. Moreover, if multiple entropy sources are used, it is natural that some of these sources are adversarial and could depend on the seed (which is hard to protect against with a dedicated attacker). Somewhat paradoxically (considering the common belief that "more entropy cannot hurt"), the mixing of such seed-dependent sources once again invalidates all the provable guarantees of seeded PRNGs, even if all the entropy is obtained from other, seed-independent sources.

We thus face a dilemma:

> *We want to support general entropy sources, for which seedless extraction is impossible, and seeded extraction is only possible under very dangerous and hard-to-ensure independence assumptions, which we would rather avoid.*

The goal of this work is to provide a meaningful solution to this dilemma, by keeping the PRNG design *seedless* while respecting properties (a) and (b) mentioned above.

1.2 *Seedless* PRNGs and Extractors from Cryptographic Hashing

We will achieve this goal by using popular *cryptographic hash functions (CHF)* as our technical tool, and by carefully defining the notion of entropy in the setting when certain components of these CHFs are assumed idealized.

Why cryptographic hashing? Before describing our solution in more detail, we explain why using CHFs appears essential for the design of *seedless*[2] PRNGs. For starters, all general-purpose software PRNGs used today, as well as all recommendations in existing PRNG standards, are based on CHFs. Hence, this setting must definitely be understood in order to provide results useful in the real world.

[2] Or, in the non-uniform setting, "seed-dependent"

However, there is a more glaring theoretical reason as well. The key component of any PRNG is the shrinking function refresh which takes the current PRNG state S as well as a new entropic input X and produces a new state $S' \leftarrow \mathsf{refresh}(S, X)$. The goal of this function is to absorb the potential entropy of X into the PRNG state S, in which case the entropy of S' should be higher than the original entropy of S. In the extraction literature, this property is called *condensing*. If one uses a seed, building such condensers is easy to accomplish information-theoretically. For example, in the PRNG design of [18], the refresh function is linear: $S' = aS + X$, where a is a seed independent of X.

In the seedless/seed-dependent setting, it is not hard to see [19] that condensers must be built cryptographically, as they require at least some form of preimage- and collision-resistance.[3] For example, when used in iteration, the simple $aS+X$ condenser function above—which yields (together with other building blocks) a provably secure seeded PRNG construction [18]—can be broken *in a catastrophic way* if the distribution of the input blocks $X_1, X_2, \ldots$ could depend on the constant a: it is not hard to see that an attacker knowing a can rather easily produce *high-entropy inputs* such that if the condenser is applied to it, the resulting would have no entropy at all. In practice, one cannot imagine a PRNG system which would risk such a catastrophic failure by critically depending on the fact that the constant a must remain hidden for the lifetime of the PRNG. Therefore, not surprisingly, all real-world PRNG designs—including those used by Windows, MacOS, and FreeBSD—critically rely on CHFs, despite lacking adequate theoretical justification.

Cryptographically secure condensers, which at an absolute minimum seedless PRNGs have to be, can be built using a (very strong form of) collision-resistance [19]. However, the types of condensers needed for applications, called average-case seedless condensers, seem to require non-standard cryptographic assumptions. For example, a relatively weak form of such average-case condensers (called "condensers for leaky sources") are already sufficient for instantiating the Fiat-Shamir heuristic for public-coin proof systems [19]—and it is a major open problem to provide such an instantiation under standard cryptographic assumptions.

To put it differently, even ignoring the fact that we want our PRNGs to be full-blown *seedless extractors*—a problem we will address next—just achieving provably secure entropy accumulation appears to require the use of CHFs as well as either (1) non-standard cryptographic assumptions (making the results appear somewhat tautologous) or (2) some supporting justification argument in an idealized model of computation, which is the approach taken by this work.

Our approach: new min-entropy notion. To describe our approach, it is instructive to recall the basic impossibility of seedless extraction for general entropy sources. Given any candidate (seedless) extractor G, an adversary can perform a so-called *extractor-fixing attack* by sampling a random input X several times until the first bit of $G(X)$ is 0. The resulting distribution X has very high entropy, but $G(X)$ is clearly not uniformly random. Observe that with a strong enough CHF G, one might be able to formally argue that the extractor-fixing attack is the "most damaging" attack possible; for example by showing, that $G(X)$ has almost full entropy (i.e., is a good condenser) for any efficiently samplable source X, as was done by [19]. In other words, using CHFs will protect against the completely devastating attacks possible with information-theoretic extractors.

However, our goal is to have a meaningful model where real randomness *extraction* is possible, so that we can later extend it all the way to the full PRNG system. Our solution will be to define a elegant and practically motivated refinement of general min-entropy *in settings where CHFs exist*, so that:

[3] For example, the ability to compute a random preimage of a given element, which is known to imply one-way functions [30], allows the attacker to produce entropic inputs whose entropy is completely lost by the refresh procedure.

(a) somewhat artificial sources resulting from intentionally performing extractor-fixing will not have much entropy according to our notion (meaning they are no longer "legitimate"); in fact, seedless extraction will become *possible* for our notion of min-entropy;

(b) most natural entropy sources, including those used by major operating systems, will likely have good entropy according to our new measure.

While our final constructions and interpretation of our security analyses will apply to real-world CHFs, such as those derived from SHA-2, SHA-3, or HKDF, at present the only rigorous way we know how to achieve our ambitious goals (a) and (b) will be by going to the idealized models of computation, such as the random oracle, the ideal cipher or the random permutation model. This is quite standard for many areas of symmetric-key cryptography, and we already indicated that doing provably secure (non-tautologous) seedless PRNG constructions in the "standard model" appears beyond our current capabilities, even for much simpler building blocks, such as (average-case) seedless condensers.

1.3 Toy Case: Monolithic Seedless Extraction from Oracle-Dependent Sources

We start by presenting our new entropy notion for the simpler problem of "monolithic randomness extraction," where the entropy source X is assumed to come in one piece (rather than slowly accumulated using a fixed-length PRNG state), and a monolithic CHF G—modeled as a monolithic random oracle—is used to output the value $R = G(X)$ (so that we temporarily ignore any find-grained structure inside G, such as Merkle-Damgård or Sponge [8] iteration).

At first, it appears that we solved our problem in a totally trivial (and uninteresting) way, even without refining standard min-entropy. Namely, in the random oracle model, the following folklore proof (see [17]) appears to show that a (seedless) random oracle G is a good extractor: For any min-entropy γ^* source X, the probability the distinguisher D can distinguish $G(X)$ is upper bounded by the probability D queries G on X, which is at most $q \cdot 2^{-\gamma^*}$, where q is the number of random oracle queries allowed to $\mathcal{A}$.[4]

Implicit in this simple proof, however, is the key assumption that the distribution X is independent of the random oracle G, meaning that our (potentially adversarial) sampler producing X is not allowed to call the random oracle G. Thus, modeling G as a random oracle is but a fancy way of introducing an exponentially long seed that is independent of the source, making extraction trivial.[5] Indeed, to capture PRNG sources X arising in the real world, we must allow the source X to depend on the ideal primitive G. For example, if the timing of computer interrupts is used as our entropy source X—which is the most common source of randomness in software PRNGs—it seems unreasonable to assume that none of these interrupts could be affected by frequent hash function computations done inside and outside the operating system.

Oracle-dependent sources. To fix this problem, in Section 3 we will explicitly model our source as part of the attacker $\mathcal{A}$, so that $\mathcal{A}^G = (\mathcal{A}_1^G, \mathcal{A}_2^G)$, where $\mathcal{A}_1^G$ outputs the *oracle-dependent* source X and passes stage Σ to the second state attacker $\mathcal{A}_2^G(\Sigma)$, whose goal is to distinguish $R = G(X)$ from uniform. Of course, for this definition to make sense, we must require that X is "legitimate,"

[4] In fact, if the length of $G(X)$ is slightly less than γ^*, we can even let $\mathcal{A}$ query all of G and use leftover-hash lemma [29] to get information-theoretic security.

[5] Prior to our work, the above modeling of sources as being independent of the ideal primitive, was the only way to overcome extractor-fixing attacks. Examples of this approach include [17, 35, 48] and many others. While these results are non-trivial due to the "non-monolithic" structure of their extractors G, none of these works model the setting where *the source could depend on the ideal primitive.*

meaning it has entropy at least γ^* given the state information Σ (for some parameter γ^*). In the standard model, this could be formalized by requiring $\mathrm{H}_\infty(X|\Sigma) \geq \gamma^*$ (see Section 2). But this is too weak, as this still allows for extractor-fixing attacks, by sampling a long random X and remembering a few bits of $G(X)$ in the leakage Σ. In fact, this extractor-fixing attack still works even if we condition on the entire random oracle G (i.e., require $\mathrm{H}_\infty(X|\Sigma, G) \geq \gamma^*$). This leads to a central question of this work:

What is the "right" notion of entropy for oracle-dependent sources X?

The key insight of our work comes from the fact that while it is reasonable to assume that the source X could *depend* on the random oracle G, the natural sources of entropy we want to extract from do not natively evaluate cryptographic hash functions, but somehow add extra entropy *in addition* to all hash function evaluations around them. For example, it is unreasonable to assume that the timing of interrupts could not depend, even slightly, on various hash function evaluations inside the computer. However, it seems that the real entropy of interrupt timings comes from the fact that the attacker cannot perfectly predict the exact lower order bits of the timing measurements, *even if the attacker knew all the hash function evaluations*. Indeed, instead of only requiring that $\mathrm{H}_\infty(X|\Sigma, G) \geq \gamma^*$, our approach will make a stronger requirement that

$$\mathrm{H}_\infty(X|(\Sigma, \mathcal{L})) \geq \gamma^* \ , \tag{1}$$

where $\mathcal{L}$ *is the* input-output *list of random oracle queries made by the sampler $\mathcal{A}_1$ to the random oracle*. Another, equivalent way to interpret this legitimacy condition is to mandate that $\mathcal{A}_1$ cannot "forget" any of its random oracle queries when passing its state Σ to $\mathcal{A}_2$, but must forget some other useful information about X, to ensure that X has entropy conditioned on Σ *and* $\mathcal{L}$.

Notice, our solution places a more stringent requirement than conditioning on the entire G, as $\mathcal{A}_1$ did not touch anything outside $\mathcal{L}$, so these un-queried values do not reduce entropy of X beyond what is done by $\mathcal{L}$. Also, when the number of queries q is not too large, the extractor-fixing is no longer a legitimate attack, since X will not have much entropy when conditioned on $\mathcal{L}$ (which contains the pair $(X, G(X))$). In fact, we can easily show *full extraction* (see Theorems 1 and 2), along the lines of the folklore proof for oracle-independent sources mentioned above. The basic intuition comes from the fact that our conditioning on the list $\mathcal{L}$ ensures that with overwhelming probability the sampler $\mathcal{A}_1$ did not himself evaluate $G(X)$, which is essential for the extractor-fixing attack to succeed.

Did we go too far? Of course, the main question is whether the legitimacy requirement $\mathrm{H}_\infty(X|(\Sigma, \mathcal{L})) \geq \gamma^*$ does not overly limit the class of high-entropy sources from which we want to extract. We believe the answer is negative. First, in the restrictive "folklore case" when X is independent of G (meaning $\mathcal{L} = \emptyset$), we get the best-possible min-entropy condition $\mathrm{H}_\infty(X|\Sigma) \geq \gamma^*$ we had in the standard (non-random-oracle) model. Namely, our notion of min-entropy relative to G includes all general min-entropy sources.[6]

Second, while we certainly allow the source X to substantially depend on G, we ensure that non-trivial bulk of entropy must come from *outside* of the actual oracle evaluation queries. In other words, while "nature," who outputs X, could conceivably be influenced by a couple of hash function evaluations, it should generate some intrinsic entropy *in addition to* (but possibly dependent on!) these evaluations. We feel that all practically used physical sources (timing if interrupts, temperature, keystroke dynamics, etc.) have very little to do with hash functions, and should easily satisfy this requirement.

[6] Of course, when we instantiate G with a real-world hash function, this is no longer the case, as we discuss below.

Thus, we believe that our technical restriction on the legitimacy for extraction using CHFs—by conditioning min-entropy on the list of hash function evaluations—strikes the right balance between allowing for seedless extraction, and yet keeping the family of high-entropy sources large and realistic for applications.

1.4 Our Results

While the above toy example (analyzed in Section 3.2) illustrated the key technical insight behind our approach, in practice it is uncommon to assume access to a monolithic random oracle G. Instead, practical hash functions are usually built from (public) compression functions, ciphers, or permutations. These underlying primitives P have limited input length and will therefore not be able to process inputs of arbitrary length m. Therefore, extractors and PRNGs should be designed in such a way that they can process short m-bit input blocks (e.g., $m = 256, 512, 1600$) and accumulate their entropy in the internal state.

Online extractors and insecurity of CBC. Thus, in Section 3.3 we formalize the more realistic notion of *online (seedless) extractors*, which slowly accumulate their long input into a fixed-length state (using access to a P), and then finalize their output once the whole input is processed. We also define both computational and information-theoretic (IT) notions of online extractor security, where in the latter notion the attacker is allowed to read the entire ideal primitive P after it finished generating the oracle-dependent source X.

Turning to natural and widely used examples of such online extractors, we show that the popular CBC mode of operation is insecure as a seedless extractor in our framework. The details of our attack are given in Section 3.4, but the result is a somewhat unexpected, since CBC is used as the extractor underlying the `CTR_DRBG` construction in the NIST PRNG standard NIST SP 800-90A Rev. 1 [4], and also as the extractor for Intel's on-chip RNG [37]. Moreover, its security was formally shown by Dodis et al. [17], but in the setting where the entropy source X was *independent* of π. In contrast, we show that once the latter assumption is relaxed to our oracle-dependent sources, the CBC extractor is no longer secure (unless one generalizes it to the Sponge construction in Section 5.3, where the input is only XORed to *part* of the state). Of course, our attack is somewhat theoretical, and does not directly translate to attacking the Intel on-chip RNG, for example. However, coupled with our positive results, we feel our attack suggests using a different online extractor, if possible.

On a positive side, in Appendix B we show several other (both computational and information-theoretic) online extractors based on popular modes of operations used inside hash functions SHA-2 and SHA-3, which are provably secure in our framework: from Merkle-Damgård with a random compression function, from Merkle-Damgård with the Davies-Meyer compression function, and from Sponges. Hence, for the first time practitioners can use seedless extractors which are both practical and have firm theoretical foundation. The security of these natural online extractors follows as special cases of more general PRNG security results, which we describe next. Due to its widespread use, we also analyze HMAC as a seedless extractor.[7] The four resulting online extractor constructions are pictorially represented in Figure 1.

Full-scale seedless PRNGs. Finally, we take all our ideas together to solve our main problem: defining and building practical, yet provably secure seedless PRNGs. In Section 4 we introduce a novel security definition for PRNGs that differs from previous notions [18, 1, 25] in several crucial ways. The detailed comparison appears in Appendix A, but we present the highlights here.

[7] Together with a variable-length PRF, the seedless HMAC extractor could then be used to build a key-derivation function (KDF), as suggested by Krawczyk [35]. A full treatment of seedless KDFs is deferred to future work, however.

First and foremost, our design is seedless. This is accomplished by carefully defining the legitimacy condition (relative to the fixed-length ideal primitive P), by conditioning our entropy notion on the list $\mathcal{L}$ of the queries to P made by the attacker. Second, our seedless design allows us to merge the "distribution sampler" and the distinguisher used by [18, 25] into a single attacker $\mathcal{A}$,[8] making our notion much simpler to describe. Third, the works of [18, 25] used a much weaker notion of worst-case min-entropy; moreover, the final entropy of the the source X was defined as sum of individual worst-case min-entropies of the individual blocks of X conditioned on all the other blocks (before and after). In contrast, we use a much better notion of *average-case* min-entropy, and only look at the global average-case min-entropy of the entire (long) vector X. Thus, our notion of entropy is much less conservative: realistic entropy sources are likely to have *much higher* entropy according to our definition, even when conditioning on the list $\mathcal{L}$. Fourth, the notion of [18, 25] had explicit "entropy estimates" that the attacker had to provide. Our notion gets rid of these estimates. Finally, and somewhat surprisingly, we still managed to define our notion of legitimacy of the entropy source in a manner which is *construction-independent*. This means that one can potentially study the entropy properties of the source in a manner independent of the PRNG used on this source.

We also define both computational and information-theoretic (IT) notions of PRNG security. As with on-line extractors, for IT-PRNGs the attacker is allowed to read the entire ideal primitive P after it finished generating the last block of it's oracle-dependent source X used for extraction. Such a notion is important for applications where privacy must hold well after the PRNG is finished its operations, or where information-theoretic security is important.

Our PRNG constructions. In Section 5 we then present three main PRNGs which are provably secure in our framework: based on Merkle-Damgård with a random compression function (see Figure 3), based on Merkle-Damgård with the Davies-Meyer compression function (see Figure 4), and based on Sponges (see Figure 5). All these constructions are extremely natural and practical, as Merkle-Damgård-based functions abstract SHA-2, while Sponges abstract SHA-3—two most widely used cryptographic hash functions. Thus, our work (including new notion of oracle-dependent entropy) could be used as theoretical justifications why these popular hash functions yield good *seedless* PRNGs (as well as online randomness extractors) even for a wide class of oracle-dependent entropy sources.

Moreover, for Merkle-Damgård based variants we also proved the security for the information-theoretic variant (the Sponge case is open, although we defined the variant which we conjecture is IT-secure). Our three computational proofs heavily use the "coefficient-H" technique [40, 13], while our two information-theoretic proofs extend the framework of so-called "graph-counting" proofs [17, 7, 24] to bound the collision probability of iterated hash constructions. One novel challenge we had to solve here comes from the input source could depend on the list $\mathcal{L}$ of the ideal primitive queries, which breaks the "source-primitive" independence assumption crucially used in these already subtle proofs.

We also showed numeric examples of how we propose to use our constructions. Overall, we believe all of them are deployment ready, and we hope this work will start influencing future PRNG deployments, and will be incorporated into next RNG standards.

Implications to standard model. To overcome the impossibility of seedless extraction, our entropy notion is defined relative to the ideal primitive P. As we argued in detail in Section 1.2, working in the idealized model seems somewhat inherent to our approach, provided we wish to

[8] Since we no longer need to hide the seed from the distribution sampler, forcing us to separate it from the attacker.

avoid highly non-standard, and likely tautological, cryptographic assumptions about the CHF we are using in the standard model. Still, it is good to ask what one might expect from our extractor and PRNG constructions with real-world CHFs, such as those based on SHA-2, SHA-3, HMAC, etc. As we already mentioned, we believe these constructions are secure for real-world entropy sources, because our idealized notion of entropy informally corresponds to sources which have fresh entropy, even given all the hash function evaluations happening around the source. To state the counter-positive, we believe that any real-world attack against our constructions with existing hash functions will either require a highly artificial entropy source, or will find a surprising weakness in the corresponding CHF.

1.5 Other Related Work

We mention some important categories of related works, in particular with respect to seedless extraction, PRNGs, and their security.

Seeded extractors and PRNGs. We already mentioned the extensive work on seeded extractors started by the seminal paper of Nisan and Zuckerman [38], and why they are problematic in our context. In the context of PRNGs, the first seeded PRNG notion was defined and constructed by Dodis et al. [18], who extended the prior "monolithic PRNG" definition of Barak and Halevi [1] (which did not explicitly talk about the seed, assuming the extraction module is "good enough" for the class of distributions produced by the entropy source). This line of work was extended in various ways by [20, 25, 28], where the latter two works were also analyzed in the random permutation model (in addition to the seed). However, none of these works considered a seedless setting for general entropy sources.

Extractors and PRNGs in ideal models. Extractors and PRNGs were also studied in the ideal models by several works [17, 9, 42, 48]. While not having explicit seeds, these works nevertheless modeled the *entropy source as being independent of the ideal primitive*. As we argued above, such oracle-independent modeling seems to be too restrictive for many realistic scenarios. Also, from a theory point of view, it effectively allows an exponentially long seed (the randomness used to sample the corresponding ideal primitive), making the positive results less interesting theoretically than the above-mentioned work on seeded extractors and PRNGs.

Indeed, the main motivation of all these papers was not to design theoretically optimal extractors and PRNGs, but to analyze the heuristic use of various cryptographic hash functions and popular modes of operations (such as CBC, HMAC, etc.) for randomness generation and extraction—a task these objects were not natively designed for. From this perspective, and given their widespread use, analyzing their extraction properties was an important first step in understanding their security, even under the restrictive oracle-independence assumption. Our work could be viewed as making a critical leap forward, by dropping—for the first time—the oracle-independence assumption, but instead carefully modeling what constitutes entropy in the much more realistic, oracle-dependent setting.

Restricting the class of entropy sources. This line of work has primarily focused on the question of extraction, by assuming that the source X has more structure beyond entropy. Early examples [46, 16, 10, 36, 14] include various bit-fixing and limited dependence sources, culminating with the question of extracting from several independent sources [2, 11]. While mathematically very elegant, the types of sources studied by these works appear "too structured" to be realistic in the PRNGs scenario.

A different kind of restriction on the entropy source was studied by Barak et al. [3]. Rather than restrict sources by some property of their distribution, the work of [3] allows for arbitrary min-entropy sources, but assumes they come from an a-priori bounded number of distributions. While potentially promising for the setting of PRNGs, there are two disadvantages of the work of [3] as compared to this work. First, the work of [3] concentrated on the "monolithic" extraction setting, and did not address the question of entropy accumulation, where the entropy in X might come slowly from a large number of samples, and has to be accumulated into bounded state. In particular, it is unclear how to extend their constructions to address entropy accumulation with a fixed-length state. Second, the particular solutions offers by [3] used so called t-wise independent hash functions for a large values of t (at least as large as the overall source length). These functions are quite inefficient, and might not be fast enough for general purpose PRNGs.

We note that our work could also be viewed as overcoming impossibility of extraction by restricting the type of the source. However, we feel that our modeling is more natural for (and, thus, applicable to) the existing entropy sources, as used by the current PRNGs.

Low-Complexity Samplers. Introduced by Trevisan and Vadhan [45] and later extended by [32], here one assumes that the entropy source producing input X is unable to run the extractor/PRNG even once, thus making it impossible to do extractor-fixing. While this might be useful for situations where the entropy source is extremely simple, it is too restrictive for most applications, such as general purpose PRNG design studied in this work. In contrast, in this work the entropy source can easily run the extractor, but the legitimacy condition is defined in a way that doing the trivial extractor-fixing attack—by running the extractor—will result in a low-entropy, "illegitimate" source.

Randomness condensers. This approach, formalized by Dodis, Ristenpart and Vadhan [19], relaxes the security guarantees of the randomness extractor to only ensure that the output of the (seedless or "source-dependent-on-seed") condenser is almost full entropy, despite not being perfectly uniform. Indeed, this weaker security turns out to be sufficient for several applications, such as key derivation schemes for signature schemes. Unfortunately, if we want an extractor rather than a condenser—which is essential for general purpose PRNGs—this approach is not sufficient.

UCEs and public-seed pseudorandomness. The notion of universal computational extractors (UCEs) [6], and its generalizations [44], study a complementary problem to the one studies here: how to extract from any entropy source which is only *computationally*-hard-to-predict, so it only has "computational entropy". On a positive, and similar to this work, when instantiated with constructions from an ideal primitive P, a UCE hash function yields a good extractor even if the inputs to it (the actual source) can be sampled depending on the ideal primitive. The issue, however, is that the current UCE notion inherently *requires a seed*, making in inapplicable for the PRNG scenario. An interesting direction for future research could be to extend our work to deal with computational entropy, by defining and constructing *seedless* UCEs in idealized models, and possibly extending them to full-blown seedless PRNGs for computational entropy.

2 Preliminaries

2.1 Statistical Distance and Min-Entropy

The *statistical distance* of two random variables X and Y is $\mathsf{SD}(X,Y) = \frac{1}{2}\sum_x |\mathsf{P}[X=x] - \mathsf{P}[Y=y]|$. The *prediction probability* of a random variable X is $\mathsf{Pred}(X) := \max_x \mathsf{P}[X=x]$, and we also denote

$\mathsf{Pred}(X|y) := \max_x \mathsf{P}[X = x|Y = y]$. The conditional version of prediction probability is defined as

$$\mathsf{Pred}(X|Y) \;:=\; \mathbf{E}_{y \leftarrow Y}\big[\mathsf{Pred}(X|y)\big] \;.$$

The *(average-case) conditional min-entropy* is $\mathrm{H}_\infty(X|Y) = -\log(\mathsf{Pred}(X|Y))$.

2.2 Security Games

All of the security properties considered in this paper are captured by considering a game between a challenger and an attacker $\mathcal{A}$, both of which may have access to an ideal primitive P. The goal of the attacker is to guess a random bit b chosen by the challenger, who offers a set of oracles to the attacker to aid with this task. The *advantage* of $\mathcal{A}$ is defined as

$$2 \cdot \Big|\; \mathsf{P}[\mathcal{A} \text{ wins}] - 1/2 \;\Big| \;,$$

where the probability is over the randomness of $\mathcal{A}$, of the challenger, and of the ideal primitive. The cases where $b = 0$ and $b = 1$ are referred to as the *real world* and the *ideal world*, respectively. One may equivalently consider $\mathcal{A}$'s advantage at telling these two worlds apart, i.e.,

$$\Big|\; \mathsf{P}[\mathcal{A} = 1|b = 0] - \mathsf{P}[\mathcal{A} = 1|b = 1] \;\Big| \;.$$

3 Seedless Extraction

As a warm-up for full-fledged seedless PRNGs, this section considers the simpler property of *extraction*, i.e., producing uniformly random bits from weak high-entropy sources. Extraction can be seen as corresponding to the post-compromise security of PRNGs, and as such it will be implied by PRNG *robustness* (as defined in Section 4.2). The definition of extraction security in Section 3.1 considers the entropy of the attacker's input to the extractor conditioned on the attackers *state* and the *queries* made to an ideal primitive P. A definition is provided for *computational* or *information-theoretic* security. IT extractors differ from computational ones in that the output of the extractor remains random even if the attacker, *after providing the input*, is given the entire function table of the underlying ideal primitive. That is, IT extractors achieve so-called *everlasting security* (cf. works in the hybrid bounded-storage model by Harnik and Naor [26]).

Section 3.2 considers extracting with a *monolithic random oracle*. The corresponding security proofs (for the computational and IT cases) are instructive for understanding the actual PRNG constructions provided in Section 5. Since considering a monolithic oracle is not motivated by any hash function used in practice, Section 3.3 introduces the concept of *online* extraction. An online extractor accumulates the entropy of its inputs in an internal state, from which uniform randomness can be produced. Finally, in order to illustrate the non-triviality of online extraction, Section 3.4 shows that extractors based on the popular CBC mode are not suitable for extraction.

3.1 Definition

In a model with idealized primitive P (chosen from some set $\mathcal{P}$), seedless extractors are algorithms $\mathsf{ext}^P : \mathcal{X} \to \mathcal{Y}$ with oracle access to P. The security definition for such extractors considers a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where both parts have access to P. The first stage $\mathcal{A}_1$ outputs a value x and some state information σ for $\mathcal{A}_2$. The second stage takes an input $y \in \mathcal{Y}$ and outputs a single bit (i.e., it acts as a distinguisher).

For an attacker $\mathcal{A}$, denote by $\mathcal{L}_1$ and $\mathcal{L}_2$ the (random variables corresponding to) the lists of the P-queries made by $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

Definition 1. *An attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is called a* q-attacker *if $|\mathcal{L}_1 \cup \mathcal{L}_2| \leq q$ always; it is called a* q-IT-attacker *if $|\mathcal{L}_1| \leq q$ always.*

That is, for IT-attackers the second stage $\mathcal{A}_2$ may make an arbitrary number of queries to P. Equivalently, $\mathcal{A}_2$ can be thought of as being given the entire function table of P.

The security game for seedless extractors in the P-model roughly requires that if the extractor is given a high-entropy input by $\mathcal{A}_1$, then $\mathcal{A}_2$ cannot tell the extractor output apart from a random value in $\mathcal{Y}$, even given the state information σ and access to P. Formally, it proceeds as follows:

1. The challenger chooses $b \leftarrow \{0,1\}$ and $P \leftarrow \mathcal{P}$ uniformly at random.
2. $\mathcal{A}_1$ gets access to P and produces $(\sigma, x) \leftarrow \mathcal{A}_1^P$.
3. The output of the extractor is computed as $y_0 \leftarrow \mathsf{ext}^P(x)$. Moreover, the challenger picks a value $y_1 \leftarrow \mathcal{Y}$ uniformly at random.
4. The second-stage attacker $\mathcal{A}_2$ is given σ and y_b and outputs a decision bit $b' \leftarrow \mathcal{A}_2^P(\sigma, y_b)$. The attacker wins if and only if $b' = b$.

The advantage of $\mathcal{A}$ in this extraction game is denoted by $\mathrm{Adv}_{\mathsf{ext}}^{\mathrm{ext},P}(\mathcal{A})$.

An attacker has to satisfy a legitimacy condition. Intuitively, this condition requires that the output X of $\mathcal{A}_1$ have high min-entropy even conditioned on the state information Σ and the list of queries $\mathcal{L}_1$.[9]

Definition 2. *An attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is said to be* γ^*-legitimate *if, in the extraction game above,*

$$\mathrm{H}_\infty(X|\Sigma\mathcal{L}_1) \;\geq\; \gamma^* \;.$$

The above finally leads to the following definition of seedless extractor in the P-model:

Definition 3. *An algorithm $\mathsf{ext}^P : \mathcal{X} \to \mathcal{Y}$ with oracle access to P is a seedless* $(\gamma^*, q, \varepsilon)$-(IT-)extractor in the P-model *if for every γ^*-legitimate q-(IT-)attacker $\mathcal{A}$,*

$$\mathrm{Adv}_{\mathsf{ext}}^{\mathrm{ext},P}(\mathcal{A}) \;\leq\; \varepsilon \;.$$

3.2 Seedless Extraction with a Monolithic Random Oracle

For instructive purposes it is useful to consider monolithic extraction, i.e., the case where the ideal primitive P itself is used as an extractor. To exemplify this, assume P is a random oracle, i.e., a function $G : \{0,1\}^m \to \{0,1\}^n$ chosen uniformly at random. Then, the monolithic extractor is defined as follows:

Construction 1 (Monolithic extractor)**.** The monolithic seedless extractor $\mathsf{mono}^G : \{0,1\}^m \to \{0,1\}^n$ using a random oracle $G : \{0,1\}^m \to \{0,1\}^n$ is defined by

$$\mathsf{mono}^G(x) := G(x) \;.$$

Theorem 1 (Monolithic seedless extraction)**.** *Construction* mono *is a* $(\gamma^*, q, \varepsilon)$*-extractor in the G-model for*

$$\varepsilon \;\leq\; \frac{q}{2^{\gamma^*}} \;.$$

[9] Note, in the extraction game the definition of $\mathcal{L}_1$ is the same in the real and the ideal worlds. For our future definitions of PRNGs, however, it will be important that the notion of legitimacy is defined in the ideal world (i.e., conditioned on $b = 1$).

The proof of Theorem 1 is deferred and can be found in Section 6.2.

Theorem 2 (Monolithic seedless IT-extraction)**.** *Construction* mono *is a* $(\gamma^*, q, \varepsilon)$-IT-extractor in the G-model *for*

$$\varepsilon \;\leq\; \frac{1}{2}\sqrt{\frac{2^{-(\gamma^*-n)}}{1-\rho}} + \rho \;,$$

where $\rho = q/2^{\gamma^*}$.

The proof of Theorem 2 is deferred and can be found in Section 7.2.

Parameter choices. In terms of concrete parameters, observe the following for the constructions towards monolithic seedless extraction from above:

- **Computational:** If we let $n = 512$ and $q = 2^{80}$. We would need $\gamma^* \approx 160$ to get 80 bits of security.
- **Information Theoretic:** We let $n = 512$. We also approximate $1/(1-\rho) \leq 2$, very generously Then, if we set for example $q = 2^{80}$. We would need the entropy loss, i.e, $\gamma^* = 160$ for 80 bits of security.

3.3 Online Extraction

In practice it is uncommon to assume access to a monolithic random oracle. Instead, practical hash functions are usually built from (public) compression functions, ciphers, or permutations. These underlying primitives P have limited input length and will therefore not be able to process inputs of arbitrary length m. Therefore, extractors (and PRNGs) should be designed in such a way that they can process short m-bit input blocks (e.g., $m = 256, 512, 1600$) and accumulate their entropy in the internal state.

An "accumulating" extractor satisfies ext satisfies the same security Definition 3, but its syntax can be thought of as two algorithms $\mathsf{ext} = (\mathsf{refresh}, \mathsf{finalize})$, where refresh accumulates entropy in an internal state and finalize produces the extractor output from the current state.

Definition 4. *An* online extractor construction *consists of two algorithms* $\mathsf{ext} = (\mathsf{refresh}, \mathsf{finalize})$*, where*

- refresh *takes a state* s *and an input* $x \in \{0,1\}^m$ *and produces a new state* $s' \leftarrow \mathsf{refresh}^P(s, x)$*, and*
- finalize *takes a state* s *and produces an output* $y \in \{0,1\}^r$*, i.e.,* $y \leftarrow \mathsf{finalize}^P(s)$.

An online extractor processing m*-bit inputs and producing* r*-bit output is called a* (m, r)-online extractor.

The security definition for online extractors additionally considers the number ℓ of times refresh is called by the attacker, i.e., it considers (q, ℓ)-attackers.

Definition 5. *An algorithm* $\mathsf{ext}^P : \mathcal{X} \to \mathcal{Y}$ *defined by two algorithms* $\mathsf{ext} = (\mathsf{refresh}, \mathsf{finalize})$ *with oracle access to* P *is an* $(\gamma^*, q, \ell, \varepsilon)$-(IT-)online extractor in the P-model *if for every* γ^**-legitimate* (q, ℓ)*-(IT-)attacker* $\mathcal{A}$*,*

$$\mathrm{Adv}^{\mathsf{ext},P}_{\mathsf{ext}}(\mathcal{A}) \;\leq\; \varepsilon \;.$$

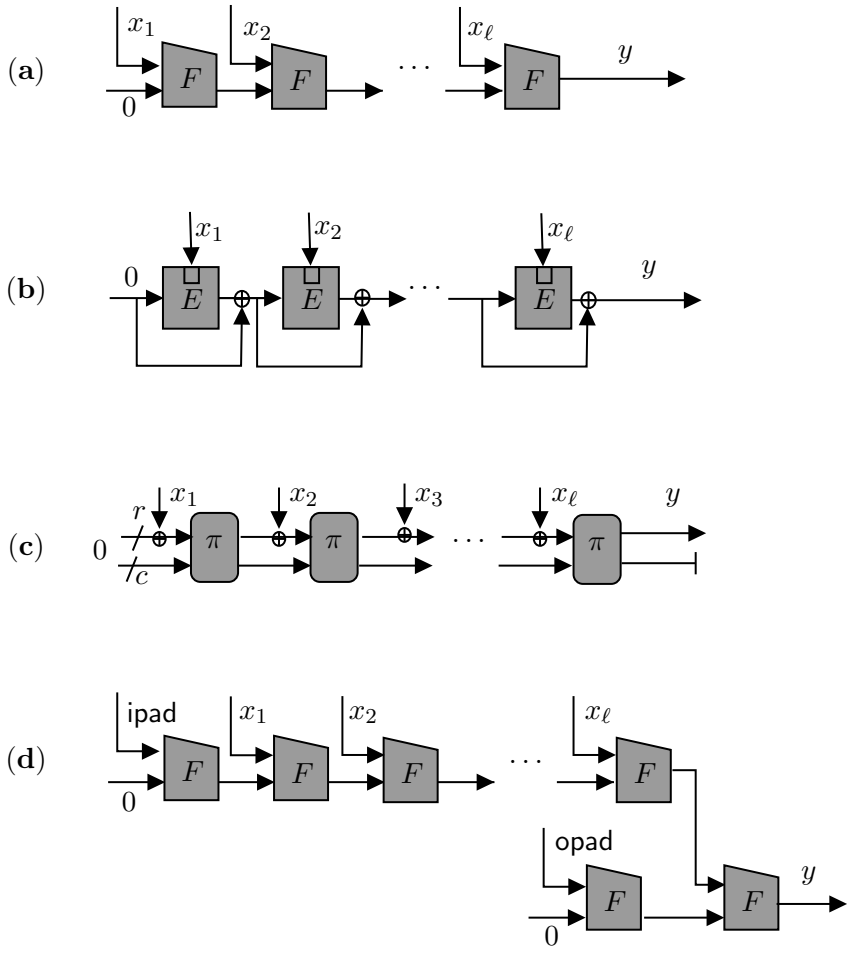

Figure 1: *Four online computational extractors are represented in this diagram. These are based on:* **(a)** *Merkle-Damgård with a random compression function;* **(b)** *Merkle-Damgård with Davies-Meyer;* **(c)** *Sponge; and* **(d)** *HMAC.*
Each extractor is shown to process inputs $x_1 \dots x_\ell$ (calls to refresh*) to compute the output y (call to* finalize*). The IT variant truncates the output y and returning the first r bits. However, note that the Sponge construction needs a truncated output even for the computational variant.*

Online extractors can be built just like the PRNG constructions in Section 5, and, in fact, the corresponding security results follow as a special case of PRNG security. Correspondingly, their treatment is deferred until Section 5, where such online extractors (and, in fact, full-fledged PRNGs) can be obtained from Merkle-Damgård with a random compression function, from Merkle-Damgård with the Davies-Meyer compression function, and from Sponges. For the reader's convenience, Appendix B contains the online extractor constructions along with the security bounds—for applications where extraction is sufficient. In addition, said section in the appendix also considers the HMAC construction as a seedless extractor. HMAC is roughly based on Merkle-Damgård, but it has a few modifications/additions that are unnecessary for extraction. However, due to its wide-spread use, we point out how to modify the Merkle-Damgård proofs to obtain a security statement for HMAC. These constructions are pictorially represented in Figure 1.

In contrast to Merkle-Damgård and Sponges, as shown in the next section, using the CBC paradigm (which can be thought of as an "extreme sponge") will not lead to a secure online extractor.

3.4 CBC-Based Extractors Are Insecure

A natural candidate for an online seedless extractor is using a permutation in CBC mode. A CBC-based extractor construction uses a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ to absorb n-bit inputs. Its refresh function is defined as

$$\mathsf{refresh}^{\pi}(s, x) \;=\; \pi(s \oplus x) \;.$$

However, it turns out that this approach does not lead to a secure extractor. This section presents a simple attack against CBC-based extractors. The attack works irrespective of how the finalization function is defined.

Theorem 3 (Attack against CBC Extractors)**.** *Let* refresh *as defined above. There exists an ℓ-legitimate q-attacker $\mathcal{A}$ with black-box access to a function* finalize*, such that for all* CBC = (refresh, finalize)

$$\mathrm{Adv}^{\mathrm{ext},\pi}_{\mathsf{CBC}}(\mathcal{A}) \;=\; 1 - 2^{-(r-1)} \;,$$

where r is the output length of the extractor, $q = 2\ell + 2\alpha$, and α is the query complexity of finalize*.*

The idea of the attack is to have the attacker create the i^{th} input block as either $\pi^i(0^n) \oplus \pi^i(1^n)$ or 0, each with probability 1/2.[10] After ℓ such steps, the attacker will have provided ℓ bits of entropy (even conditioned on its π-queries), but only a single bit will have accumulated in the state, which will be $\pi^i(0^n)$ or $\pi^i(1^n)$, each with probability 1/2.

Proof. Consider a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{A}_1$ works as follows:

1. Initially, set $a = 0^n$ and $b = 1^n$.
2. For blocks $i = 1$ to ℓ:
   (a) Set $x_{i,0} = 0$ and $x_{i,1} = a \oplus b$. Choose random bit $\beta \leftarrow \{0,1\}$ and set $x_i = x_{i,\beta}$.
   (b) Set $a \leftarrow \pi(a)$ and $b \leftarrow \pi(b)$.
3. Set state information to $\sigma \leftarrow (a, b)$. In particular, *forget* all values $x_{i,b}$ and all values of β. Return σ and $x = (x_1, \ldots, x_\ell)$.

$\mathcal{A}_2$, given $\sigma = (a, b)$ and y as input, proceeds as follows:

1. Compute finalize(a) and finalize(b). If either of them equals y, return 0; else, return 1.

Assume the initial state of the extractor is 0^n. In order to understand the attack, consider the state of the extractor after XORing the first input x_1: it is either 0^n or 1^n, each with probability 1/2. After applying refresh, which consists of just applying π to the state, the new state is either $\pi(0^n)$ or $\pi(1^n)$, again with probability 1/2 each.

The second input x_2 is either 0^n or $\pi(0^n) \oplus \pi(1^n)$; XORing it to the state results in either $\pi(0^n)$ or $\pi(1^n)$. After applying refresh, the new state is $\pi^2(0^n)$ or $\pi^2(1^n)$. Extending this argument to all ℓ inputs, the state of the extractor after absorbing them is either $a = \pi^\ell(0^n)$ or $a = \pi^\ell(1^n)$. After calling finalize, the state is either finalize(a) or finalize(b). Therefore, in the real world ($b = 0$), $\mathcal{A}_2$ will always output 0, whereas in the ideal world ($b = 1$) this will only happen with probability at most $2^{-(r-1)}$.

In terms of efficiency, observe that $\mathcal{A}_1$ makes 2ℓ queries to π, and $\mathcal{A}_2$ makes twice the number α of π-queries required to evaluate a call to finalize.

Finally, in order to show that $\mathcal{A}$ is legitimate, observe first that the state σ can be computed from the queries made by $\mathcal{A}_1$. It easily seen that given only the queries, the vector $x = (x_1, \ldots, x_\ell)$ has ℓ bits of entropy. □

[10] Here, π^i denotes the i-fold application of π.

4 Pseudorandom Number Generators with Input

A *pseudorandom number generator with input (PRNG)* is a stateful cryptographic primitive. It gradually accumulates entropy in its state by absorbing inputs and can be used to output pseudorandom bits once the entropy of the state is sufficiently high. Moreover, it is both forward and backward secure, i.e., past outputs remain random upon future state compromise, and, by absorbing sufficient amounts of entropy, a PRNG can recover from state compromise.

This section introduces a novel security definition for PRNGs that differs from previous notions in several crucial ways. Specifically, a comparison to the original *robustness* notion by Dodis *et al.* [18], based on work by Barak and Halevi [1], as well as to an adaptation of it by Gazi and Tessaro [25] for idealized models is provided in Appendix A.

This paper considers two notions of PRNGs: computational PRNGs and information-theoretically secure (IT) PRNGs. IT PRNGs differ from computational PRNGs in that once the attacker stops interacting with the PRNG, the output of the PRNG remains random even if the attacker is given the entire function table of the underlying ideal primitive. That is, IT PRNGs achieve so-called *everlasting security* (cf. works in the hybrid bounded-storage model by Harnik and Naor [26]). This distinction is analogous to that between seedless extractors and IT seedless extractors (cf. Section 3).

4.1 Syntax

A PRNG consists of two algorithms: one for absorbing new inputs and one for producing pseudorandom outputs. Formally, it is defined as follows:

Definition 6 (Syntax of PRNGs)**.** *A* pseudorandom number generator with input (PRNG) *is a pair of algorithms* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *having access to an ideal primitive* P *and sharing an* n*-bit state* s*, where*

- $\mathsf{refresh}$ *takes a state* s *and an input* $x \in \{0,1\}^m$ *and produces a new state* $s' \leftarrow \mathsf{refresh}^P(s, x)$*, and*
- next *takes a state* s *and produces a new state and an output* $y \in \{0,1\}^r$*, i.e,* $(s', y) \leftarrow \mathsf{next}^P(s)$*.*

A PRNG processing m*-bit inputs and producing* r*-bit output is called a* (m, r)-PRNG*.*

4.2 Security Game

Robustness game. PRNGs are expected to satisfy the so-called *robustness* property, which captures the properties discussed at the beginning of Section 4. The corresponding security game is depicted in Figure 2. The game initially chooses a random bit b and initializes the state of the PRNG to 0^n. Subsequently, it offers the following oracles to $\mathcal{A}$:

- **adv-refresh**(x) calls the $\mathsf{refresh}$ procedure to absorb $x \in \{0,1\}^n$ into the internal state of the PRNG;
- **get-next** and **get-next*** allow the attacker to get pseudorandom outputs by calling the next procedure on the current state and returning the output y. The difference between the two oracles is that **get-next** is supposed to be called only when the state has high entropy, whereas **get-next*** can be called *prematurely*, i.e., before the state has absorbed enough randomness for the next function to output pseudorandom values (cf. definition of legitimate attackers below).

The PRNG Robustness Game

init
- $s \leftarrow 0^n$
- $b \leftarrow \{0,1\}$

adv-refresh (x)
- $s \leftarrow \mathsf{refresh}^P(s, x)$

get-state
- **return** s

next-ror
- $(s, y_0) \leftarrow \mathsf{next}^P(s)$
- $y_1 \leftarrow \{0,1\}^r$
- **return** y_b

get-next/get-next*
- $(s, y) \leftarrow \mathsf{next}^P(s)$
- **return** y

set-state (s^*)
- $s \leftarrow s^*$

Figure 2: *Oracles for the PRNG robustness game.*

- **next-ror** works like the **get-next**-oracles, except that it creates a challenge, i.e., if $b = 1$, it outputs a uniform random value $y_1 \in \{0,1\}^r$ instead of the PRNG output y_0.
- **get-state** and **set-state** model state compromises by letting the attacker learn the current state or set it to an arbitrary value, respectively.

The advantage of $\mathcal{A}$ in the robustness game is denoted by $\mathrm{Adv}^{\mathrm{rob},P}_{\mathsf{PRNG}}(\mathcal{A})$.

Canonical attackers. It will be useful to define to following notion of canonical attackers: Consider the interaction of an attacker $\mathcal{A}$ with the robustness game. The following events are called *entropy drains*:

- the beginning of the game,
- calls to **get-state** or **set-state**, and
- calls to **get-next***.

In other words, entropy drains are the events that cause the PRNG state to lose its entropy, which includes premature calls to next. An attacker $\mathcal{A}$ is said to be *canonical* if it does not make **get-next*** queries nor the following query pattern: an entropy drain followed by one or more **adv-refresh** queries, followed by a **get-state** query.

Considering canonical attackers only is without loss of generality. This is because the above sequence of queries can be simulated by the attacker by making a **get-state** query right away and computing the output of **get-state** or **get-next*** itself. In particular, for every attacker $\mathcal{A}$, there exists a canonical attacker $\mathcal{A}$ with the same advantage. All attackers in the remainder of this work are therefore assumed to be canonical.

Legitimate attackers. In order to obtain a sensible definition devoid of trivial attacks, attackers must satisfy a "legitimacy" condition. The condition roughly requires that an attacker only ask for challenges when it has sufficient amount of uncertainty about the PRNG's internal state.

Towards formalizing the legitimacy condition, consider the interaction of $\mathcal{A}$ with a *variant* of the robustness game defined as follows: Whenever oracles **next-ror** or **get-next** are called, instead of evaluating next, the game simply uses two uniformly random and independent values (s, y) as the output of next.

Observe that this variant of the robustness game, called the *legitimacy game* corresponds to an interaction between $\mathcal{A}$ and an *ideal* PRNG, which produces perfect randomness. Moreover, the legitimacy game is *construction-independent.*

In the legitimacy game, define now the following random variables immediately before $\mathcal{A}$ makes the i^{th} call to oracle **get-next** or **next-ror**:

- $\mathcal{L}_i$: the list of P-queries by $\mathcal{A}$ and the corresponding answers;
- Σ_i: the state of $\mathcal{A}$;
- $\overline{X}_i$: vector of inputs provided by $\mathcal{A}$ since the *the most recent entropy drain (MRED)*; and
- S_i: the state of the PRNG immediately after the MRED.

The legitimacy condition requires that $\mathcal{A}$ provide inputs that have high min-entropy even conditioned on its current state, the queries so far, and the state of the PRNG after the MRED.

Definition 7 (Legitimate attackers)**.** *An attacker $\mathcal{A}$ is said to be* γ^*-legitimate *if for all i,*

$$\mathrm{H}_\infty(\overline{X}_i|\Sigma_i\mathcal{L}_i S_i) \;\geq\; \gamma^* \;,$$

where MREDs are defined as above.

In order to capture IT-legitimate attackers (against IT PRNGS), the set of entropy drains is extended to include

- calls to **get-next** and **next-ror**.

With this definition of MRED and notation analogous to that in the previous definition, IT-legitimate attackers are defined as follows:

Definition 8 (Legitimate IT attackers)**.** *An attacker $\mathcal{A}$ is said to be* γ^*-IT-legitimate *if for all i,*

$$\mathrm{H}_\infty(\overline{X}_i|\Sigma_i\mathcal{L}_i S_i) \;\geq\; \gamma^* \;,$$

w.r.t. the extended definition of MRED.

Robust PRNGs. We are now ready to quantify the efficiency of attacker $\mathcal{A}$, and to define our final notion of PRNG robustness.

Definition 9 (Attacker efficiency)**.** *An attacker is called a* (q, t, ℓ)-attacker *if*

- *q is the maximum number of P-queries it makes,*
- *ℓ is the maximum number of* **adv-refresh** *calls between any entropy drain and successive call to either* **next-ror** *or* **get-next***, and*
- *t is the maximum total number of calls to any oracle in the robustness game other than* **adv-refresh**.

An attacker is called a (q, t, ℓ)-IT-attacker *if it satisfies the above conditions but makes an arbitrary number of queries to P* after *the interaction with the challenger ends.*

Definition 10 (Robustness of PRNGs)**.** *A PRNG construction* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *with oracle access an ideal primitive P is* $(\gamma^*, q, t, \ell, \varepsilon)$-(IT-)robust *in the P-model if for every γ^*-(IT-)legitimate (q, t, ℓ)-(IT-)attacker,*

$$\mathrm{Adv}^{\mathrm{rob},P}_{\mathsf{PRNG}}(\mathcal{A}) \;\leq\; \varepsilon \;.$$

Observe that online extractors (cf. Definition 4) are a special case of robust PRNGs. In terms of construction, the PRNG next algorithm can be replaced by $\mathsf{finalize}$, which simply discards the state output by next. If then the PRNG robustness game is relaxed such that the only queries the attacker can make are (a) arbitrarily many queries to **adv-refresh** followed by (b) $t = 1$ query to **next-ror**, one obtains a notion equivalent to Definiton 3.

5 Constructions of PRNGs

This section presents three simple, intuitive, and—most importantly—practical PRNG constructions:

- a construction based on the *Merkle-Damgård paradigm* using a public *fixed-length compression function*;
- a construction based on the *Merkle-Damgård paradigm* using the *Davies-Meyer compression function* (as in SHA-2), which is built from any public block cipher; and
- a construction based on the *Sponge paradigm* (as in SHA-3), which uses a public permutation.

For PRNGs based on the MD paradigm, there are in fact two constructions: one achieving normal, computational PRNG security and one achieving information-theoretic (IT) security. The security analyses of all three constructions are provided in Sections 6 (computational PNRGs) and 7 (IT security). There is also an IT candidate based on Sponges, but its security analysis is left for future work.

For the reader's convenience, Appendix B states the corresponding online extractor constructions along with the security bounds—for applications where extraction is sufficient.

5.1 PRNGs from Merkle-Damgård

A PRNG can be obtained from a compression function F as follows (cf. Figure 3):[11]

Construction 2 (PRNG from Merkle-Damgård)**.** The (m, r)-PRNG construction $\mathsf{MD} = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is defined as follows:[12]

- $\mathsf{refresh}^F(s, x) = F(s, x)$, and
- $\mathsf{next}^F(s) = (F(s, 0), F(s, 1) \| \cdots \| F(s, r/n))$.

The security of Construction 2 is proved in the F-model, where F is a uniformly random function. The proof of the following theorem is deferred to Section 6.4.

Theorem 4 (Robustness of Merkle-Damgård PRNGs)**.** *Construction 2 is a* $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$*-robust PRNG in the* F*-model for*

$$\varepsilon_{\text{rob}} \;\leq\; 2t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} \right),$$

where $\tilde{q} = q + r/n + 1$.

An IT-robust PRNG based on Merkle-Damgård can be obtained if the next function simply outputs the truncated state (and outputs 0^n as the new state):

Construction 3 (IT-PRNG from Merkle-Damgård)**.** The (m, r)-PRNG construction $\mathsf{MD}_r = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is defined as follows:

- $\mathsf{refresh}^F(s, x) = F(s, x)$, and

[11] To reduce notational clutter, the algorithms refresh and next of the PRNG constructions are not "branded" with the design name. There will be no ambiguity as to which construction is meant in any place in this paper.

[12] The integer arguments to the compression function are to be naturally mapped to $\{0,1\}^n$.

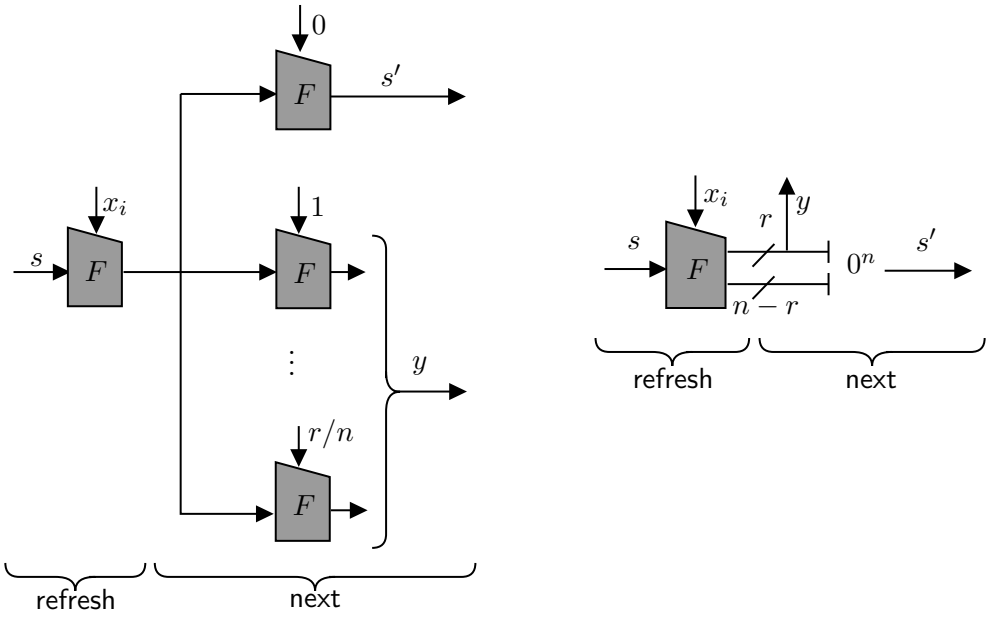

Figure 3: *Procedures* refresh *(processing a single-block input x_i) and* next *of Merkle-Damgård PRNG constructions with compression function F. Left: Computationally secure Construction 2; right: IT secure Construction 3.*

- $\mathsf{next}^F(s) = (0^n, s[1..r])$.

The security of Construction 3 is proved in the F-model, where F is a uniformly random function. To state the theorem for the IT construction, for an integer ℓ, let

$$d'(\ell) = \max_{\ell' \in \{1,\ldots,\ell\}} \left|\{d \in \mathbb{N} \ : \ d|\ell'\}\right| .$$

Observe that, asymptotically, $d'(\ell)$ grows very slowly, i.e., as $\ell^{o(1)}$. Furthermore, let F be a random compression function. The proof of the following theorem is deferred to Section 7.4.

Theorem 5 (IT-Robustness of Merkle-Damgård PRNGs)**.** *Construction 3 is a $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-IT-robust PRNG in the F-model, where*

$$\varepsilon_{\text{rob-it}} \ \leq \ \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 \cdot \frac{\tilde{q}^2 2^r}{2^{2n}}} \ + \ t\rho \ ,$$

for $\rho = \frac{\tilde{q}^2}{2^r}$ where $\tilde{q} = q + t\ell$.

Parameter choices. In terms of concrete parameters, observe the following for the Merkle-Damgård constructions above:

- **Computational PRNG:** If one were to use SHA-512 as compression function with $n = 512$, and, moreover, choose $r = n$. We let $t = 1, q = 2^{80}$ and let $\gamma^* = \ell$. This assumes that we get at least one bit of entropy from each block. We would need $\gamma^* \approx 160$ to get 80 bits of security.

- **IT PRNG:** For example, assume SHA-512's compression function is used, i.e., $n = 512$. If we let $r = 256$, then we get (we also approximate $1/(1-\rho) \leq 2$, very generously)

$$\varepsilon_{\text{rob-it}} \;\leq\; \frac{t}{2}\sqrt{2^{257-\gamma^*} + \frac{\ell \cdot d'(\ell)}{2^{256}}} \;+\; t\frac{q^2}{2^{256}} \;,$$

  We let $\ell = \gamma^*$. Then, if we set for example $q = 2^{80}$. We would need the entropy loss, i.e, $\gamma^* - r = 162$ for 80 bits of security.

5.2 PRNGs from Merkle-Damgård with Davies-Meyer

The Davies-Meyer compression function maps two inputs $a \in \{0,1\}^m$ and $b \in \{0,1\}^n$ to an n-bit string

$$E(b,a) \oplus a \;,$$

where E is an arbitrary block cipher (where b is the key and a the input).[13] Correspondingly, a PRNG can be obtained from E as follows (cf. Figure 4):

Construction 4 (PRNG from MD-DM)**.** The (n,r)-PRNG construction $\mathsf{DM} = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with Davies-Meyer (MD-DM) uses a cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and is defined as follows:[14]

- $\mathsf{refresh}^E(s,x) = E(x,s) \oplus s$, and
- $\mathsf{next}^E(s) = (E(0,s) \oplus s, E(1,s) \oplus s \| \cdots \| E(r/n,s) \oplus s)$.

The security of Construction 4 is proved in the E-model, where E is a cipher chosen uniformly at random from the set of all ciphers and can be queried in both the forward and backward direction. The proof of the following theorem is deferred to Section 6.5.

Theorem 6 (Robustness of MD-DM PRNGs)**.** *Construction 4 is a $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-robust PRNG in the E-model for*

$$\varepsilon_{\text{rob}} \;\leq\; 4t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}}\right),$$

where $\tilde{q} = q + r/n + 1$.

In the IT-secure variant of the MD-DM construction, $\mathsf{refresh}$ remains the same, but next will truncate the input state to r bits, which it outputs, and then zero out the state.

Construction 5 (IT-PRNG from MD-DM)**.** The (n,r)-PRNG construction $\mathsf{DM}_r = (\mathsf{refresh}, \mathsf{next})$ using Merkle-Damgård with Davies-Meyer (MD-DM) uses a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and is defined as follows:

- $\mathsf{refresh}^E(s,x) = E(x,s) \oplus s$, and
- $\mathsf{next}^E(s) = (0^n, s[1..r])$.

The security of Construction 5 is proved in the E-model, where E is a cipher chosen uniformly at random from the set of all ciphers and can be queried in both the forward and backward direction. Let $d'(\ell)$ be defined as in Section 5.1. The proof of the following theorem is deferred to Section 7.5.

[13] A (block) cipher is an efficiently computable and invertible permutation $E(k,\cdot) : \{0,1\}^n \to \{0,1\}^n$ for every key $k \in \{0,1\}^n$.

[14] The integer arguments to the cipher are to be naturally mapped to $\{0,1\}^n$.

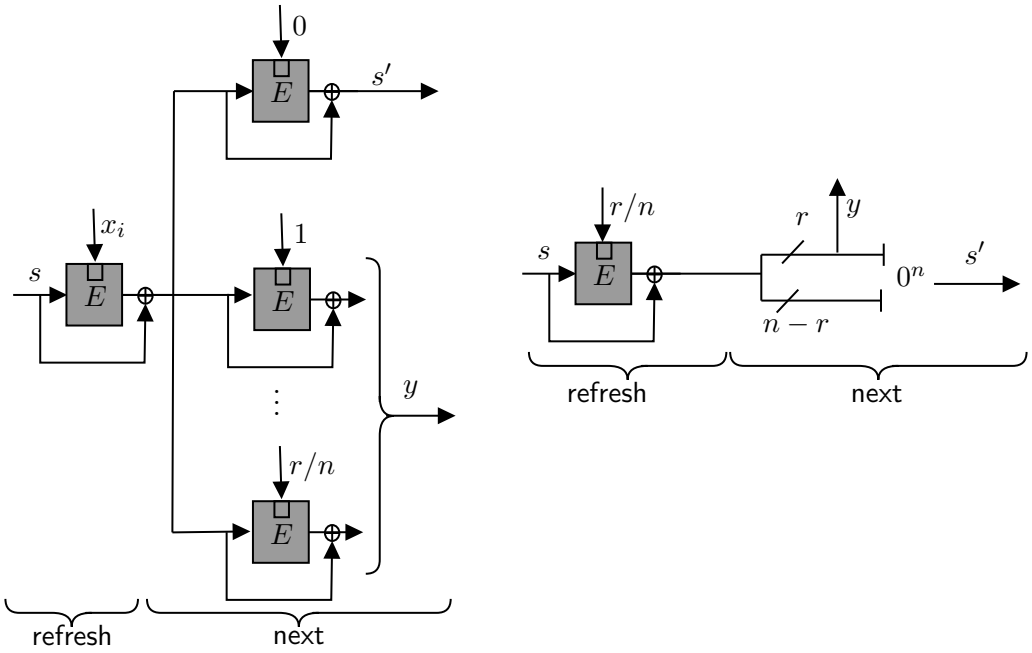

Figure 4: *Procedures* refresh *(processing a single-block input x_i) and* next *of Merkle-Damgård PRNG constructions with the Davies-Meyer compression function based on a block cipher E. Left: Computationally secure Construction 4; right: IT secure Construction 5.*

Theorem 7 (IT-Robustness of MD-DM PRNGs)**.** *Construction 5 is a $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-IT-robust PRNG in the E-model, where*

$$\varepsilon_{\text{rob-it}} \;\leq\; \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + 64\ell^4 \cdot \frac{2^r}{2^{2n-2}} + 16\ell^2\tilde{q}^2 \cdot \frac{2^r}{2^{2n-2}}} + t\rho \;,$$

for $\rho = \frac{\tilde{q}^2}{2^r}$ where $\tilde{q} = q + t\ell$

Parameter choices. In terms of concrete parameters, observe the following for the PRNG constructions from Merkle-Damgård with Davies-Meyer above:

- **Computational PRNG:** SHA-512 is a 512-bit block cipher algorithm that encrypts 512 bit hash value using the input as key. Therefore, we let $n = 512$ and set $r = n$. We let $t = 1, q = 2^{80}$ and let $\ell = \gamma^*$. This assumes that we get at least one bit of entropy from each block. We would need $\gamma^* \approx 163$ to get 80 bits of security.
- **IT PRNG:** We again let $n = 512$. If we let $r = 256$, then we get (we also approximate $1/(1-\rho) \leq 2$, very generously)

  $$\varepsilon_{\text{rob-it}} \;\leq\; \frac{t}{2}\sqrt{2^{129-\gamma^*} + \frac{\ell \cdot d'(\ell)}{2^{127}}} \;+\; t\frac{q^2}{2^{128}} \;,$$

  We let $\ell = \gamma^*$. Then, if we set for example $q = 2^{80}$. We would need the entropy loss, i.e, $\gamma^* - r = 162$ for 80 bits of security.

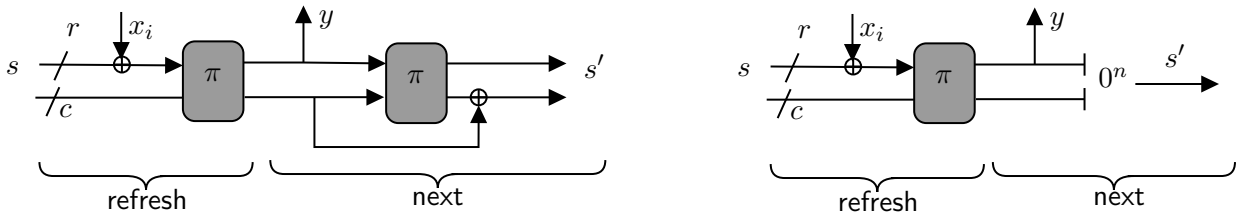

Figure 5: *Procedures* refresh *(processing a single-block input x_i) and* next *of Merkle-Damgård PRNG constructions with compression function F. Left: Computationally secure Construction 2; right: IT candidate Construction 3.*

5.3 PRNGs from Sponges

Let $n \in \mathbb{N}$ and $n = r + c$. In the following, for an n-bit string s, let $s = s^{(\mathsf{r})} \| s^{(\mathsf{c})}$ be decomposition of s into an r-bit and c-bit string. A PRNG using the Sponge paradigm can be obtained from a permutation π as follows (cf. Figure 5):

Construction 6 (PRNG from Sponges)**.** The Sponge-based PRNG construction $\mathsf{Spg} = (\mathsf{refresh}, \mathsf{next})$ uses a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ to absorb and produce r-bit inputs and outputs, respectively, and is defined as follows:

- $\mathsf{refresh}^\pi(s, x) = \pi(s \oplus x \| 0^c)$, and
- $\mathsf{next}^\pi(s) = (\pi(s) \oplus 0^r \| s^{(\mathsf{c})}, s^{(\mathsf{r})})$.

The next function design is due to Hutchinson [27], who simplified a proposal by Gazi and Tessaro [25]. Recall that the Merkle-Damgård constructions have a "parallel" next function in order to produce r/n blocks of random output with $r/n+1$ calls to the ideal primitive, where the additional call is used to produce a new state. Were it not for this optimization, on order to obtain r bits of output, one would have to apply the next function r/n times in a row, which would results in twice the number of ideal-primitive calls.

The next function for Sponges, on the other hand, only makes a single call to the ideal primitive to produce both a new state and the random output. Therefore, no parallel next function is provided for the Sponge-based PRNG.

The security of Construction 6 is proved in the π-model, where π is a uniformly random permutation, which can be queried in both the forward and backward direction. The proof of the following theorem is deferred to Section 6.6.

Theorem 8 (Robustness of Sponge PRNGs)**.** *Construction 6 is a $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-robust PRNG in the π-model for*

$$\varepsilon_{\text{rob}} \;\leq\; 4t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} + \frac{\tilde{q}^2}{2^c} \right),$$

where $\tilde{q} = q + r/n + 1$.

Observe that the bound in Theorem 8 is only reasonable when c is large enough, which matches the fact that CBC-based PRNGs—which correspond to the case $c = 0$, are not secure.

In the IT variant of the Sponge construction, refresh remains the same, but next will truncate the input state to r bits, which it outputs, and then zero out the state.

Construction 7 (IT PRNG from Sponges)**.** The Sponge-based PRNG construction $\mathsf{Spg}_r = (\mathsf{refresh}, \mathsf{next})$ uses a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ to absorb and produce r-bit inputs and outputs, respectively, and is defined as follows:

- $\mathsf{refresh}^\pi(s, x) = \pi(s \oplus x \| 0^c)$, and
- $\mathsf{next}^\pi(s) = (0^n, s[1..r])$.

Proving the security of the IT PRNG based on the Sponge paradigm, Construction 7, is left as an interesting open problem.

Parameter choices. In terms of concrete parameters, observe the following for the PRNG constructions from Sponges above: above:

- **Computational PRNG:** SHA-3 like parameters have $n = 1600$ and $c = 1024$. We let $t = 1, q = 2^{80}$ and let $\ell = \gamma^*$. This assumes that we get at least one bit of entropy from each block. We would need $\gamma^* \approx 163$ to get 80 bits of security.

6 Security Proofs for Computational Constructions

This section analyzes all previously presented computationally secure constructions. The main technique in all proofs is the so-called H-coefficient technique, which is discussed first, in Section 6.1. Section 6.2 then shows the security of the monolithic extractor from Section 3.2. While rather straight-forward, the corresponding proof is quite instructive for the security proofs of the PRNG constructions from Section 5. For these proofs, it is convenient to define and establish a number of intermediate properties, which are discussed in Section 6.3. Finally, Sections 6.4 to 6.6 establish the robustness (cf. Section 4.2) of the PRNG constructions.

6.1 The H-Coefficient Technique

When considering an adaptive distinguisher $\mathcal{D}$ that tries to tell apart two different worlds, usually termed *real* and *ideal* experiments, the H-coefficient technique [40] is a handy tool for analyzing the distinguishing advantage of $\mathcal{D}$.

The H-coefficient technique considers transcripts between distinguisher $\mathcal{D}$ and the challenger. These transcripts are partitioned into two groups: the *good* transcripts and the *bad* transcripts. For good transcripts τ, an H-coefficient proof will commonly derive a lower bound on the ratio of the probability of τ occurring in the real world and that of τ occurring in the ideal world. For bad transcripts, which are normally defined as the transcripts for which said lower bound cannot be derived, one upper bounds the probability that they occur. This latter bound can be proved in the *ideal* world, which usually greatly simplifies the derivation.

Transcripts. The interaction of $\mathcal{D}$ with either the real or the ideal experiment produces a transcript T that contains the queries made by $\mathcal{D}$ and the corresponding answers given by the challenger. For a fixed transcript τ, denote by $\mathsf{p}_0(\tau)$ and $\mathsf{p}_1(\tau)$ the probabilities that the real and ideal experiments, respectively, the challenger produces the answers in τ if asked the queries in τ.[15] Similarly, the behavior of a distinguisher $\mathcal{D}$ is described by a function $\mathsf{p}_\mathcal{D}(\tau)$ that assigns to τ the probability that $\mathcal{D}$ produces the queries in τ if given the answers in τ. Observe that, therefore, the probability of a particular transcript τ occurring in an interaction of $\mathcal{D}$ with the real experiment

[15] Observe that $\mathsf{p}_0(\tau)$ and $\mathsf{p}_1(\tau)$ depend only on the corresponding experiment and are independent of $\mathcal{D}$.

is $\mathsf{P}[T_0 = \tau] = \mathsf{p}_{\mathcal{D}}(\tau) \cdot \mathsf{p}_0(\tau)$ and, similarly, $\mathsf{P}[T_1 = \tau] = \mathsf{p}_{\mathcal{D}}(\tau) \cdot \mathsf{p}_1(\tau)$ for the ideal world. In the following, denote by $\mathcal{T}$ the set of all transcripts τ.

Bounding the distinguishing advantage. The distinguishing advantage of $\mathcal{D}$ is upper bounded by the statistical distance

$$\begin{aligned}
\mathsf{SD}(T_0, T_1) &= \sum_{\tau \in \mathcal{T}} \max\{0, \mathsf{P}[T_1 = \tau] - \mathsf{P}[T_0 = \tau]\} \\
&= \sum_{\tau \in \mathcal{T}} \max\{0, \mathsf{p}_{\mathcal{D}}(\tau) \cdot \mathsf{p}_1(\tau) - \mathsf{p}_{\mathcal{D}}(\tau) \cdot \mathsf{p}_0(\tau)\} \\
&= \sum_{\tau \in \mathcal{T}} \mathsf{p}_{\mathcal{D}}(\tau) \cdot \mathsf{p}_1(\tau) \left(1 - \frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)}\right) \\
&= \sum_{\tau \in \mathcal{T}} \mathsf{P}[T_1 = \tau] \left(1 - \frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)}\right) . \qquad (2)
\end{aligned}$$

Suppose that for some set $\Gamma \subseteq \mathcal{T}$ of *good* transcripts, a lower bound

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \geq 1 - \varepsilon$$

is known for all $\tau \in \Gamma$ and some $\varepsilon \geq 0$. Then, (2) becomes

$$\begin{aligned}
\sum_{\tau \in \mathcal{T}} \mathsf{P}[T_1 = \tau] \left(1 - \frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)}\right) &\leq \sum_{\tau \in \Gamma} \left(1 - \frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)}\right) + \sum_{\tau \in \mathcal{T} \setminus \Gamma} \mathsf{P}[T_1 = \tau] \\
&\leq \varepsilon + \mathsf{P}[T_1 \in \mathcal{T} \setminus \Gamma] ,
\end{aligned}$$

where transcripts $\tau \in \mathcal{T} \setminus \Gamma$ are commonly referred to as *bad* transcripts. Given the above, applying the H-coefficient technique entails defining a set of good transcripts, bounding the fraction above, and showing that bad transcripts are unlikely. Note that the latter can be done in the ideal experiment, which is usually considerably easier than doing so in the real experiment.

Theorem 9 (H-coefficient method)**.** *For two experiments described by* $\mathsf{p}_0(\cdot)$ *and* $\mathsf{p}_1(\cdot)$*, respectively, if there exists a set* $\Gamma \subseteq \mathcal{T}$ *and* $\varepsilon, \delta \geq 0$ *satisfying*

1. ***(ratio analysis)*** $\mathsf{p}_0(\tau)/\mathsf{p}_1(\tau) \geq 1 - \varepsilon$ *for all* $\tau \in \Gamma$ *and*
2. ***(bad event analysis)*** $\mathsf{P}[T_1 \notin \Gamma] \leq \delta$*,*

then the distinguishing advantage of any distinguisher $\mathcal{D}$ *is bounded by* $\varepsilon + \delta$*.*

6.2 Monolithic Extractor

From Section 3.2, recall the monolithic extractor mono defined to work with a random oracle $F : \{0,1\}^m \to \{0,1\}^n$:

Construction 1 (Monolithic extractor)**.** The monolithic seedless extractor $\mathsf{mono}^G : \{0,1\}^m \to \{0,1\}^n$ using a random oracle $G : \{0,1\}^m \to \{0,1\}^n$ is defined by

$$\mathsf{mono}^G(x) := G(x) .$$

Construction 1 is a seedless extractor according to Definition 3.

Theorem 1 (Monolithic seedless extraction)**.** *Construction* mono *is a* $(\gamma^*, q, \varepsilon)$*-extractor in the G-model for*

$$\varepsilon \;\leq\; \frac{q}{2^{\gamma^*}} \;.$$

The proof of Theorem 1 is a straight-forward application of the H-coefficient technique. The idea is to first show that unless $\mathcal{A}_1$ or $\mathcal{A}_2$ queries the input x provided by $\mathcal{A}_1$, the real and ideal worlds (i.e., the cases where $b = 0$ and $b = 1$, respectively) are indistinguishable. That is, the corresponding ratio of transcript probabilities is 1. Transcripts where x is in the query list are defined to be *bad* transcripts, and the second part of the proof shows that bad transcripts are unlikely to occur due to the legitimacy of $\mathcal{A}$. The latter proof crucially relies on the fact that the H-coefficient technique enables performing the bad-event analysis in the *ideal* world.

Proof. Consider a transcript of the interaction between an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and the challenger of the extraction game (as defined in Section 3.1). It consists of

- the input x provided by $\mathcal{A}_1$,
- the value y^* output by the game (which is either the output y_0 of the extractor on x or a uniformly random value from $\{0,1\}^n$), and
- the query/answer list $L = L_1 \cup L_2$ of $\mathcal{A}_1$'s and $\mathcal{A}_2$'s interaction with F.

That is $\tau = (y^*, x, L)$. A *bad* transcript occurs when $\mathcal{A}_1$ or $\mathcal{A}_2$ queries F at input x, i.e., when L contains a pair of the form $(x, *)$. In order to apply Theorem 9, one merely needs to bound the probability ratio for good transcripts (Lemma 10) and the probability of a bad transcript occurring in the ideal world, i.e, for $b = 1$ (Lemma 11). □

Lemma 10 (Ratio analysis)**.** *For all* good *transcripts* τ*,*

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \;=\; 1 \;.$$

Proof. Fix a good transcript τ and consider first $\mathsf{p}_1(\tau)$. Since in the ideal world y^* is sampled uniformly,

$$\mathsf{p}_1(\tau) \;=\; p_L \cdot 2^{-n} \;,$$

where p_L denotes the probability that a uniform random function is consistent with the queries in L. In the real world,

$$\mathsf{p}_0(\tau) \;=\; p_L \cdot q_\tau \;,$$

where q_τ is the probability that $F_L(x) = y^*$ over a function F_L that is sampled uniformly at random conditioned on being consistent with L. Since τ is a good transcript, F_L is not constrained by L at coordinate x, and, hence, $q_\tau = 2^{-n}$. □

Remark 1. In the above proof, note that p_L does not include the probability that x appears in the transcript. Recall from Section 6.1 that the behaviors $\mathsf{p}(\tau)$ are the probabilities that the experiment produces the answers in τ *when given* the queries in τ (and the value x is a query by the distinguisher).

Lemma 11 (Bad event analysis)**.** *For the set* $\mathcal{B}$ *of bad transcripts (as defined above),*

$$\mathsf{P}[T_1 \in \mathcal{B}] \;\leq\; \frac{q}{2^{\gamma^*}} \;.$$

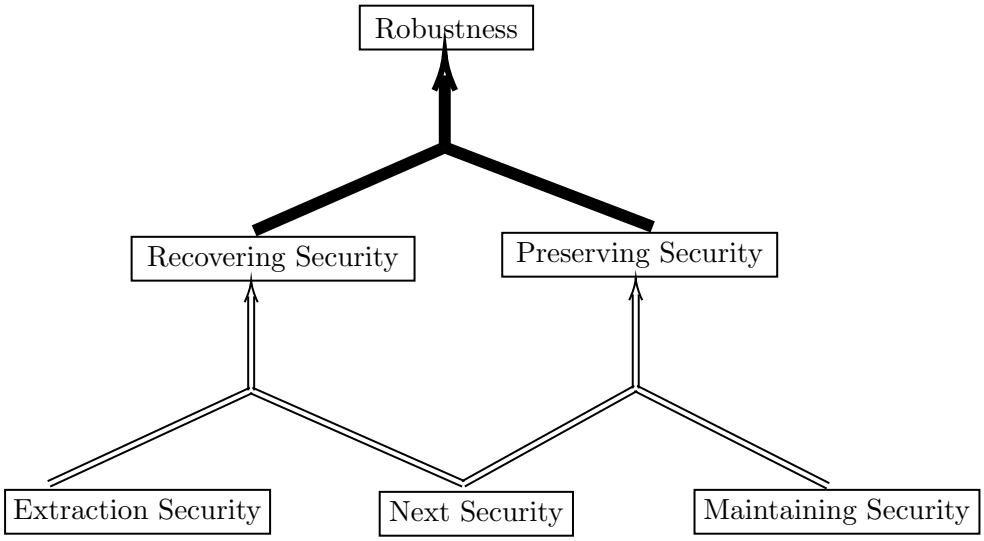

Figure 6: *This figure represents the implication relations between the different intermediate notions of security. The filled arrows stand for a generic proof, while the unfilled arrows represent a construction-specific proof.*

Proof. Recall that by the γ^*-legitimacy of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\mathrm{H}_\infty(X|\Sigma\mathcal{L}_1) \;\geq\; \gamma^* \;.$$

Observe that in the ideal world, the output of the extraction game is a uniformly random value Y^*, which is independent of the input X produced by $\mathcal{A}_1$. The sampling order of the ideal experiment can therefore be changed to be the following:

1. Sample F uniformly at random.
2. Run $\mathcal{A}_1$ until it outputs σ and x, thereby also generating the list of queries L_1.
3. Choose y^* uniformly at random.
4. Run $\mathcal{A}_2$ on input (σ, y^*), letting it make additional queries L_2.
5. Resample the input X conditioned on $(\Sigma, \mathcal{L}_1) = (\sigma, L_1)$.

Note that since the conditioning includes $\mathcal{L}_1$, $\mathcal{A}_1$ makes the same queries, L_1, during the first run and the resampling process. Moreover, since conditioned on the values of $(\Sigma, \mathcal{L}_1)$, X and $\mathcal{L}_2$ are independent, the min-entropy condition holds for $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$, the list of all queries made by $\mathcal{A}$ during the experiment, as well. That is,

$$\mathrm{H}_\infty(X|\Sigma\mathcal{L}) \;\geq\; \gamma^* \;.$$

Thus, the probability that the resampled input X is contained in any query in the list L is at most $q \cdot 2^{-\gamma^*}$. □

The PRNG Recovering Game

init
  $b \leftarrow \{0,1\}$

chall $(s_0, x_1, \ldots, x_\ell)$
  for $i = 1, \ldots, \ell$
    $s_i \leftarrow \mathsf{refresh}(s_{i-1}, x_i)$
  if $b = 0$
    return $\mathsf{next}(s_\ell)$
  else
    $(s, y) \leftarrow \{0,1\}^{n+r}$
    return (s, y)

Figure 7: *Oracles for PRNG recovering game.*

6.3 Intermediate PRNG Security Notions

In keeping with tradition in PRNG literature, it is useful to define two simple properties called *recovering* and *preserving*. Recovering security requires that that if after an entropy drain, sufficient amount of entropy has been absorbed into the PRNG state, the output of the next function look random. Preserving security asks that after absorbing adversarially chosen inputs, a high-entropy state not become compromised, and the output of the next function after the absorption look random.

Recovering and preserving security can be shown to generically (i.e., for any PRNG construction) imply robustness. In order to establish these two properties themselves, it helps to introduce three further properties called *extraction*, *maintaining*, and *next security*:

- **Extraction security:** show that the PRNG *state* is indistinguishable from a uniform one after sufficient amounts of entropy have been absorbed;
- **Maintaining security:** show that the PRNG *state* is indistinguishable from a uniform one if it is random initially and arbitrary inputs are absorbed;
- **Next security:** show that the output of the next function is indistinguishable from a random value if it is called on a random input.

For each PRNG construction considered in this work, extraction and next security imply recovering security, and maintaining and next security imply preserving security. These proofs, however, are *not* generic and must be repeated for each PRNG construction. Figure 6 illustrates these implications.

6.3.1 Recovering and Preserving Security

As stated above, it is useful to define two simple properties called *recovering* and *preserving*, which together generically imply robustness via a hybrid argument.

Recovering security. The intuition behind Recovering security is that if after an entropy drain, sufficient amount of entropy (from the perspective of the attacker) has been absorbed into the state, then the output of next is indistinguishable from a uniformly random value in $\{0,1\}^{n+r}$.

The corresponding game is depicted in Figure 7. It lets the attacker specify an initial state s_0 and a vector of inputs $x_1, \ldots, x_\ell$; the inputs are then absorbed one-by-one, and next is called on the resulting state. The game returns the output of next if $b = 0$ and a uniform value if $b = 1$. The advantage of an attacker $\mathcal{A}$ in the recovering game is denoted by $\mathrm{Adv}^{\mathrm{rec},P}_{\mathsf{PRNG}}(\mathcal{A})$.

The PRNG Preserving Game

init
- $s_0 \leftarrow \{0,1\}^n$
- $b \leftarrow \{0,1\}$

chall $(x_1, \ldots, x_\ell)$
- **for** $i = 1, \ldots, \ell$
  - $s_i \leftarrow \mathsf{refresh}(s_{i-1}, x_i)$
- **if** $b = 0$
  - **return** $\mathsf{next}(s_\ell)$
- **else**
  - $(s, y) \leftarrow \{0,1\}^{n+r}$
  - **return** (s, y)

Figure 8: *Oracles for PRNG preserving game.*

Similarly to the robustness game, an attacker has to satisfy a legitimacy condition. In particular, $\mathcal{A}$ is γ^*-legitimate if

$$\mathrm{H}_\infty(X_1, \ldots, X_\ell | \Sigma \mathcal{L} S_0) \;\geq\; \gamma^* \;,$$

where

- Σ is the state of $\mathcal{A}$ just before the call to **chall**,
- $\mathcal{L}$ is the list of query and answers $\mathcal{A}$ has made to P up to the call to **chall**, and
- S_0 is the initial state that the adversary provides.

For the recovering game, q again denotes the maximum number of P-queries that $\mathcal{A}$ makes, and ℓ is the maximum number of blocks with which it calls oracle **chall**; a corresponding attacker is referred to as (q, ℓ)-*attacker*.

Definition 11. *A PRNG construction* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *is said to be* $(\gamma^*, q, \ell, \varepsilon)$-recovering *in the P-model if for every γ^*-legitimate (q, ℓ)-attacker,*

$$\mathrm{Adv}^{\mathrm{rec},P}_{\mathsf{PRNG}}(\mathcal{A}) \;\leq\; \varepsilon \;.$$

Preserving security. At a high level, preserving security requires that by absorbing adversarially chosen inputs, a high-entropy state cannot become compromised, and the output of next after the absorption is indistinguishable from a uniformly random value in $\{0,1\}^{n+r}$.

The corresponding game is depicted in Figure 8. It lets the attacker specify a vector of inputs $x_1, \ldots, x_\ell$; the inputs are then absorbed into a *randomly chosen* state one-by-one, and next is called on the resulting state. The game returns the output of next if $b = 0$ and a uniform value if $b = 1$. The advantage of an attacker $\mathcal{A}$ in the preserving game is denoted by $\mathrm{Adv}^{\mathrm{pre},P}_{\mathsf{PRNG}}(\mathcal{A})$. There is no legitimacy constraint on $\mathcal{A}$; the parameters q and ℓ are defined as before for a (q, ℓ)-attacker.

Definition 12. *A PRNG construction* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *is said to be* (q, ℓ, ε)-preserving *in the P-model if for every (q, ℓ)-attacker,*

$$\mathrm{Adv}^{\mathrm{pre},P}_{\mathsf{PRNG}}(\mathcal{A}) \;\leq\; \varepsilon \;.$$

Recovering and preserving imply robustness. As mentioned above, in order to establish robustness of a PRNG construction, it suffices to prove that it is both *recovering* and *preserving*.

Theorem 12. *Consider a PRNG construction* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *for which* refresh *makes α P-calls and* next *makes β P-calls. Furthermore, assume* PRNG *is both*

- $(\gamma^*, q, \ell, \varepsilon_{\text{rec}})$-*recovering and*
- $(q, \ell, \varepsilon_{\text{pre}})$-*preserving*

in the P-model. Then, PRNG *is also $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-robust in the P-model, where*

$$\varepsilon_{\text{rob}} \;\leq\; t \cdot (\varepsilon_{\text{rec}} + \varepsilon_{\text{pre}}) \;.$$

The proof of Theorem 12 is provided in Appendix C.1.

6.3.2 Extraction, Maintaining, and Next Security

Extraction security. Extraction security requires that after absorbing blocks with high joint entropy, the state of the PRNG be indistinguishable from a uniformly random one. The corresponding game is a variant of the game for recovering security (cf. Figure 7) in which next is not applied to s_ℓ; instead, s_ℓ or a uniformly random value is output, depending on whether $b = 0$ or $b = 1$.

The legitimacy of an attacker $\mathcal{A}$ as well as parameters q and ℓ are defined identically to the recovering game (cf. Section 4.2); the advantage of $\mathcal{A}$ against extraction security of PRNG in the P-model is denoted by $\text{Adv}^{\text{ext},P}_{\mathsf{PRNG}}(\mathcal{A})$.

Maintaining security. Maintaining security is a variant of the preserving game (cf. Figure 7) in which next is not applied to s_ℓ; instead, s_ℓ or a uniformly random value is output, depending on whether $b = 0$ or $b = 1$. The parameters q and ℓ are defined identically to the preserving game (cf. Section 4.2); the advantage of $\mathcal{A}$ against maintaining security of PRNG in the P-model is denoted by $\text{Adv}^{\text{mtn},P}_{\mathsf{PRNG}}(\mathcal{A})$.

Next security. Next security requires that the output of the next function on a uniformly random state be indistinguishable from a uniformly random string. That is, an attacker $\mathcal{A}$, making at most q queries to the ideal primitive P, tries to distinguish $\mathsf{next}^P(S)$ from U_{n+r} for a uniformly random $S \in \{0,1\}^n$. Denote by $\text{Adv}^{\text{next},P}_{\mathsf{PRNG}}(\mathcal{A})$ the advantage of $\mathcal{A}$.

6.4 PRNGs from Merkle-Damgård

This section establishes the robustness of the PRNG construction $\mathsf{MD} = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with a random compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. Recall that the construction is defined as follows (cf. Figure 3):

Construction 2 (PRNG from Merkle-Damgård)**.** The (m, r)-PRNG construction $\mathsf{MD} = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is defined as follows:[16]

- $\mathsf{refresh}^F(s, x) = F(s, x)$, and
- $\mathsf{next}^F(s) = (F(s, 0), F(s, 1) \| \cdots \| F(s, r/n))$.

Note that iteratively absorbing some input blocks $x_1, \ldots, x_\ell$ via refresh, starting with a state s_0 is identical to applying the Merkle-Damgård construction to the input with initialization vector (IV) s_0, which is denoted by $\mathsf{MD}^F_{s_0}(x_1, \ldots, x_\ell)$ in the remainder of this section.

[16] The integer arguments to the compression function are to be naturally mapped to $\{0,1\}^n$.

Theorem 4 (Robustness of Merkle-Damgård PRNGs)**.** *Construction 2 is a* $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$*-robust PRNG in the* F*-model for*

$$\varepsilon_{\text{rob}} \;\leq\; 2t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} \right),$$

where $\tilde{q} = q + r/n + 1$.

As outlined in Section 6.3, the idea of the proof is to first establish extraction, maintaining, as well as next security, and to then show that extraction and next security imply recovering security and that maintaining and next security imply preserving security. More precisely, the proof proceeds as follows:

- **Extraction security:** In order to establish extraction security using the H-coefficient method (cf. Theorem 9 in Section 6.1), one first defines *bad* transcripts as transcripts for which the attacker has queried F on all coordinates needed to evaluate the PRNG on the inputs it provided.

  For good transcripts, there is at least one missing F query to obtain the output of the PRNG. The probability ratio is lower bounded by arguing that the output of the PRNG looks random to the attacker unless there is a collision among the remaining F queries.

  The probability of bad transcripts is upper bounded in the *ideal* world ($b = 1$). The proof uses the legitimacy of the attacker in a resampling argument to argue that it is unlikely that the attacker makes all the queries necessary to evaluate the PRNG on the inputs it provides. Special care has to be taken to handle collisions in the F queries.

- **Maintaining security:** The corresponding proof also employs the H-coefficient method, but is considerably more straightforward. A *bad* transcript is a transcript for which the initial (random) state of the PRNG appears in the F-queries. The ratio is bounded analogously to the extraction proof, and bounding the probability of a bad transcript is trivial.

- **Next security:** This is again a simple H-coefficient proof, which amounts to showing that unless the attacker queries F on an input $(s, *)$, where s is the (random) state to which next is applied, the output of next looks random.

- **Recovering security:** Showing that the Merkle-Damgård construction achieves recovering security is a simple hybrid argument: First, one uses extraction security to argue that the state after absorbing the inputs can be replaced by a random value. Second, using next security, one argues that the output of next on said random value looks random.

- **Preserving security:** Proving preserving security is also a simple hybrid argument: First, one uses maintaining security to argue that the state after absorbing the inputs can be replaced by a random value. Second, using next security, one argues that the output of next on said random value looks random.

The final bound in Theorem 4 follows by applying Theorem 12.

6.4.1 Extraction Security

Lemma 13 (Extraction security)**.** *The advantage of any* γ^**-legitimate* (q, ℓ)*-attacker* $\mathcal{A}$ *against extraction security of* MD *is bounded by*

$$\text{Adv}^{\text{ext},F}_{\mathsf{MD}}(\mathcal{A}) \;\leq\; \varepsilon^{\text{ext}}_{\mathsf{MD}}(\gamma^*, q, \ell) \;:=\; \frac{q^2 + q\ell + \ell^2}{2^n} + \frac{2q}{2^{\gamma^*}} \;.$$

Proof. In order to prove Lemma 13 using the H-coefficient method, consider a transcript[17]

$$\tau \;=\; (s^*, s_0, x_1, \ldots, x_\ell, L)$$

of the interaction between the $\mathcal{A}$ and the extraction game, where s^* is the value returned by the game, L is the set of queries to the oracle made by the adversary $\mathcal{A}$, and where s_0 is the initial state and $x_1, \ldots, x_\ell$ the inputs provided by $\mathcal{A}$. Let $\ell' \geq 0$ be maximal such that

$$((y_{i-1}, x_i), y_i) \in L$$

for some values $y_0, y_1, \ldots, y_{\ell'}$ with $y_0 = s_0$. A transcript τ is called a *bad transcript* if $\ell' = \ell$; otherwise, τ is called *good*.

In order to apply Theorem 9, one merely needs to bound the probability ratio for good transcripts (Lemma 14) and the probability of a bad transcript occurring in the ideal world, i.e, for $b = 1$ (Lemma 15). □

Lemma 14 (Ratio analysis)**.** *For all* good *transcripts* τ*,*

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \;\geq\; 1 - \frac{q\ell + \ell^2}{2^n} \;.$$

Proof. Fix a good transcript τ and consider first $\mathsf{p}_1(\tau)$. Since in the ideal world s^* is sampled uniformly,

$$\mathsf{p}_1(\tau) \;=\; p_L \cdot 2^{-n} \;,$$

where p_L denotes the probability that a uniform random function is consistent with the queries in L. In the real world,

$$\mathsf{p}_0(\tau) \;=\; p_L \cdot q_\tau \;,$$

where q_τ is the probability that $\mathsf{MD}_{s_0}^{F_L}(x_1, x_2, \ldots, x_\ell) = s^*$ over a function F_L that is sampled uniformly at random conditioned on being consistent with L.

It remains to derive a lower bound on q_τ. To that end, observe that due to τ being a good transcript, $\ell' < \ell$. Hence, q_τ is the probability (over F_L) that

$$Y_\ell \;:=\; \mathsf{MD}_{y_{\ell'}}^{F_L}(x_{\ell'+1}, \ldots, x_\ell) \;=\; s^* \;.$$

Consider the intermediate chaining values $Y_{\ell'+1}, \ldots, Y_{\ell-1}$ and, for $i = \ell' + 1, \ldots, \ell - 1$, define the event $\mathsf{FRESH_i}$ that Y_i is *fresh*, i.e., there is no query $((Y_i, *), *) \in L$ and $Y_i \neq Y_j$ for $j < i$. Let,

$$\mathsf{FRESH} \;:=\; \bigcap_{i=\ell'+1}^{\ell-1} \mathsf{FRESH_i} \;.$$

Then,

$$\mathsf{P}[Y_\ell = s^* | \mathsf{FRESH}] \;=\; 2^{-n}$$

since the conditioning implies that $F(Y_{\ell-1}, x_\ell)$ is freshly sampled and, hence, Y_ℓ is a uniformly random value. Moreover,

$$\mathsf{P}\left[\overline{\mathsf{FRESH_i}} \;\middle|\; \bigcap_{k=\ell'+1}^{i-1} \mathsf{FRESH_k}\right] \;\leq\; \frac{q+\ell}{2^n}$$

[17] In order to keep notation simple, ℓ—here and in the following—is the number of inputs in a particular τ, not the upper bound from Lemma 13.

as there are at most $q+\ell$ non-fresh values by the time Y_i is sampled. Here, the notation $\overline{A}$ denotes the complement of an event A. Therefore,

$$\mathsf{P}[\mathsf{FRESH}] \;\geq\; 1 - \frac{q\ell + \ell^2}{2^n} \;,$$

and, finally,

$$\begin{aligned} q_\tau \;&\geq\; \mathsf{P}[\mathsf{FRESH}] \cdot \mathsf{P}[Y_\ell = s^* | \mathsf{FRESH}] \\ &\geq\; \left(1 - \frac{q\ell + \ell^2}{2^n}\right) \cdot 2^{-n} \;, \end{aligned}$$

which implies

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \;\geq\; 1 - \frac{q\ell + \ell^2}{2^n} \;.$$

□

Lemma 15 (Bad event analysis)**.** *For the set $\mathcal{B}$ of bad transcripts (as defined above),*

$$\mathsf{P}[T_1 \in \mathcal{B}] \;\leq\; \frac{2q}{2^{\gamma^*}} + \frac{q^2}{2^n} \;.$$

Proof. Observe that in the ideal world, the output of the extraction game is a uniformly random value s^*, which is independent of the initial state S_0 and the inputs $X_1, \ldots, X_\ell$. The sampling order of the ideal experiment can therefore be changed to be the following:

1. Sample F uniformly at random.
2. Run $\mathcal{A}$ until it outputs $(\tilde{s}_0, \tilde{x}_1, \ldots, \tilde{x}_{\tilde{\ell}})$, thereby also generating the state σ and the list of queries L_0 before the challenge.
3. Choose s^* uniformly at random.
4. Continue running $\mathcal{A}$ on σ and s^*, letting it make additional queries L_1.
5. Resample the inputs $(X_1, \ldots, X_\ell)$ conditioned on $(\Sigma, \mathcal{L}_0, S_0) = (\sigma, L_0, s_0)$.

Remember that $\mathcal{L}$ is the random variable for the set of queries made while L is a particular value that the random variable takes. Observe that since the conditioning includes $\mathcal{L}_0$, $\mathcal{A}$ makes the same queries, L_0, during the first run and the resampling process. Moreover, since conditioned on $(\Sigma, \mathcal{L}_0, S_0)$, $(X_1, \ldots, X_\ell)$ and $\mathcal{L}_1$ are independent, the min-entropy condition holds for $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}_1$, the list of all queries made by $\mathcal{A}$ during the experiment, as well. That is,

$$\mathrm{H}_\infty(X_1, \ldots, X_\ell | \Sigma \mathcal{L} S_0) \;\geq\; \gamma^* \;.$$

Let $\mathcal{E}$ be the event that there exists a collision in $\mathcal{L}$, i.e., two or more queries have the same answer. Note that

$$\mathsf{P}[T_1 \in \mathcal{B}] \;\leq\; \mathsf{P}[T_1 \in \mathcal{B} | \mathcal{E}] + \mathsf{P}[\overline{\mathcal{E}}] \;\leq\; \mathsf{P}[T_1 \in \mathcal{B} | \mathcal{E}] + \frac{q^2}{2^n} \;.$$

Towards bounding $\mathsf{P}[T_1 \in \mathcal{B} | \mathcal{E}]$, consider now a particular triple $z = (\sigma, L, s_0) \in \mathcal{E}$, which is shorthand for L being collision-free. Define a *potential chain* as values $y_0, y_1, \ldots, y_\ell$ for *some* ℓ such that $y_0 = s_0$ and

$$((y_{i-1}, v_i), y_i) \in L \;.$$

for $i = 1, \ldots, \ell$ and some values $v_1, \ldots, v_\ell$. Observe that without collisions, L can contain at most q potential chains. Clearly, conditioned on $Z = z$, $T_1 \in \mathcal{B}$ if and only if for some potential chain, $X_i = v_i$ for all $i = 1, \ldots, \ell$. Hence, by the legitimacy of $\mathcal{A}$,

$$\mathsf{P}[T_1 \in \mathcal{B} | Z = z] \ \leq \ q \cdot p_z \ ,$$

where $p_z := \mathsf{Pred}(\overline{X} | Z = z)$. In expectation,

$$\begin{aligned}
\mathsf{P}[T_1 \in \mathcal{B} | \mathcal{E}] \ &= \ \sum_{z \in \mathcal{E}} \mathsf{P}[Z = z | \mathcal{E}] \cdot \mathsf{P}[T_1 \in \mathcal{B} | Z = z] \\
&\leq \ \sum_{z \in \mathcal{E}} \mathsf{P}[Z = z | \mathcal{E}] \cdot q \cdot p_z \\
&= \ q \cdot \mathsf{Pred}(\overline{X} | Z\mathcal{E}) \\
&\leq \ q \cdot \frac{\mathsf{Pred}(\overline{X} | Z)}{1 - q^2/2^n} \\
&\leq \ \frac{q}{(1 - q^2/2^n) \cdot 2^{\gamma^*}} \ \leq \ \frac{2q}{2^{\gamma^*}} \ ,
\end{aligned}$$

using that[18] $q^2/2^n \leq 1/2$ and where the penultimate inequality is due to

$$\begin{aligned}
\mathsf{Pred}(X | Z) \ &\geq \ \sum_{z \in \mathcal{E}} \mathsf{P}[Z = z] \cdot p_z \\
&= \ \mathsf{P}[\mathcal{E}] \cdot \sum_{z \in \mathcal{E}} \frac{\mathsf{P}[Z = z]}{\mathsf{P}[\mathcal{E}]} \cdot p_z \\
&= \ \mathsf{P}[\mathcal{E}] \cdot \mathsf{Pred}(\overline{X} | Z\mathcal{E}) \ .
\end{aligned}$$

□

6.4.2 Maintaining Security

Lemma 16 (Maintaining security)**.** *The advantage of any (q, ℓ)-attacker $\mathcal{A}$ against maintaining security is bounded by*

$$\mathrm{Adv}^{\mathrm{mtn}, F}_{\mathsf{MD}}(\mathcal{A}) \ \leq \ \varepsilon^{\mathrm{mtn}}_{\mathsf{MD}}(q, \ell) \ := \ \frac{q\ell + \ell^2}{2^n} + \frac{q}{2^n} \ .$$

Proof. To bound the advantage of an attacker $\mathcal{A}$ at guessing b via an H-coefficient proof, consider a transcript

$$\tau = (s^*, s_0, x_1, \ldots, x_\ell, L)$$

between $\mathcal{A}$ and the maintaining game, where s^* is the output of the game and L are the queries made by $\mathcal{A}$. A transcript is *bad* if there is a query of the type $((s_0, *), *) \in L$ and *good* otherwise. The probability of a bad transcript occurring in the $b = 1$ case is clearly at most $|L|/2^n$. Moreover, for good transcripts τ,

$$\mathsf{p}_1(\tau) \ = \ 2^{-2n} \cdot p_L \ ,$$

where p_L denotes the probability that a uniform random function is consistent with the queries in L. Furthermore, by an argument similar to that in the proof of Lemma 14,

$$\mathsf{p}_0(\tau) \ \geq \ \left(1 - \frac{q\ell + \ell^2}{2^n}\right) \cdot 2^{-2n} \cdot p_L \ .$$

The lemma follows by applying Theorem 9. □

[18] This can always be assumed as the bound would otherwise be vacuous.

6.4.3 Next Security

Lemma 17 (Next security)**.** *The advantage of any q-attacker $\mathcal{A}$ against next security is bounded by*

$$\mathrm{Adv}^{\mathrm{next},F}_{\mathsf{MD}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q) \;:=\; \frac{q}{2^n}\ .$$

Proof. For a straight-forward H-coefficient proof, consider the transcript

$$\tau = (s, s^*, L)\ ,$$

where s is the initial state, s^* is the input given to $\mathcal{A}$, and L are the queries $\mathcal{A}$ makes to F. A transcript is *bad* if L contains a query of the form $((s,*),*)$ and *good* otherwise. It is easily seen that for good transcripts, the probability ratio is 1, whereas, in the ideal world, where $\mathcal{A}$'s view is completely independent of S, the probability of a bad event is at most $q/2^n$. □

6.4.4 Recovering Security

In the following, let $\varepsilon^{\mathrm{ext}}_{\mathsf{MD}}(\gamma^*, q, \ell)$ and $\varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q)$ be as in Lemmas 13 and 17. Extraction and next security together imply recovering security:

Lemma 18 (Recovering Security)**.** *For every γ^*-legitimate (q,ℓ)-attacker $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{rec},F}_{\mathsf{MD}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{ext}}_{\mathsf{MD}}(\gamma^*, q + r/n + 1, \ell) + \varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q + r/n + 1)\ .$$

Proof. For $b \in \{0,1\}$, denote by H_b the recovering experiment conditioned on the secret bit having the value b. Moreover, define a hybrid experiment $H_{\frac{1}{2}}$ in which the challenge oracle returns $\mathsf{next}^F(U_n)$ to $\mathcal{A}$. By the triangle inequality, to prove the lemma, it suffices to bound the distance between experiments H_0 and $H_{\frac{1}{2}}$ and $H_{\frac{1}{2}}$ and H_1.

- Towards bounding the distance between H_0 and $H_{\frac{1}{2}}$, consider the following attacker $\mathcal{A}_{\mathrm{ext}}$ against extraction security of MD: $\mathcal{A}_{\mathrm{ext}}$ runs $\mathcal{A}$ answering its oracle queries by passing queries to F. At some point, $\mathcal{A}$ outputs $(s_0, x_1, ..., x_\ell)$. $\mathcal{A}_{\mathrm{ext}}$ forwards $(s_0, x_1, x_2, ..., x_\ell)$ to the challenger. $\mathcal{A}_{\mathrm{ext}}$ receives s^* as response from the challenger. $\mathcal{A}_{\mathrm{ext}}$ now computes next on the input s^*, by making $r/n+1$ additional queries to the primitive, on the inputs (s^*, i) for $i = 0, 1, \ldots, r/n$. It forwards to $\mathcal{A}$ the values $(F(s^*,0), F(s^*,1)||\ldots||F(s^*, r/n))$. It proceeds to respond to $\mathcal{A}$'s oracle calls as before and waits for $\mathcal{A}$'s guess bit. It merely forwards the same bit as its guess to the challenger.

  When the challenger's bit is 0, $\mathcal{A}_{\mathrm{ext}}$ perfectly simulates towards $\mathcal{A}$ the distribution H_0. When the challenger's bit is 1, $\mathcal{A}$ is given $\mathsf{next}(U_n)$. This corresponds to the hybrid distribution $H_{\frac{1}{2}}$. Moreover, it is easily seen that if $\mathcal{A}$ is γ^*-legitimate, so is $\mathcal{A}_{\mathrm{ext}}$. Thus, the advantage of $\mathcal{A}$ in distinguishing hybrids H_0 and $H_{\frac{1}{2}}$ is upper-bounded by the advantage of $\mathcal{A}_{\mathrm{ext}}$ in the extraction game which is $\varepsilon^{\mathrm{ext}}_{\mathsf{MD}}(\gamma^*, q + r/n + 1, \ell)$.

- Towards bounding the distance between $H_{\frac{1}{2}}$ and H_1, consider the following attacker $\mathcal{A}_{\mathrm{next}}$ against next security of MD: $\mathcal{A}_{\mathrm{next}}$ runs $\mathcal{A}$ answering its oracle queries by passing queries to F. When $\mathcal{A}$ outputs $(s_0, x_1, ..., x_\ell)$, $\mathcal{A}_{\mathrm{next}}$ returns its own distinguishing challenge, continuing to answer oracle queries for $\mathcal{A}$. In the end, it outputs $\mathcal{A}$'s guess bit.

  It is straight-forward to verify that $\mathcal{A}_{\mathrm{next}}$ perfectly simulates $H_{\frac{1}{2}}$ to $\mathcal{A}$ when given $\mathsf{next}^F(U_n)$ and H_1 when given U_{n+r}. Thus, the advantage of $\mathcal{A}$ in distinguishing hybrids $H_{\frac{1}{2}}$ and H_1 is upper-bounded by the advantage of $\mathcal{A}_{\mathrm{next}}$ in the next game, which is $\varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q + r/n + 1)$.

□

6.4.5 Preserving Security

In the following, let $\varepsilon^{\mathrm{mtn}}_{\mathsf{MD}}(q,\ell)$ and $\varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q)$ be as in Lemmas 16 and 17. Maintaining and next security together imply preserving security:

Lemma 19 (Preserving Security)**.** *For every (q,ℓ)-attacker $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{pre},F}_{\mathsf{MD}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{mtn}}_{\mathsf{MD}}(q+r/n+1,\ell) + \varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q+r/n+1)\ .$$

Proof. For $b \in \{0,1\}$, denote by H_b the preserving experiment conditioned on the secret bit having the value b. Moreover, define a hybrid experiment $H_{\frac{1}{2}}$ in which the challenge oracle returns $\mathsf{next}^F(U_n)$ to $\mathcal{A}$. By the triangle inequality, to prove the lemma, it suffices to bound the distance between experiments H_0 and $H_{\frac{1}{2}}$ and $H_{\frac{1}{2}}$ and H_1.

- Towards bounding the distance between H_0 and $H_{\frac{1}{2}}$, consider the following attacker $\mathcal{A}_{\mathrm{mtn}}$ against maintaining security of MD: $\mathcal{A}_{\mathrm{mtn}}$ runs $\mathcal{A}$ answering its oracle queries by passing queries to F. At some point, $\mathcal{A}$ outputs $(x_1,....,x_\ell)$. $\mathcal{A}_{\mathrm{mtn}}$ forwards $(x_1,x_2,...,x_\ell)$ to the challenger. $\mathcal{A}_{\mathrm{mtn}}$ receives s^* as response from the challenger. $\mathcal{A}_{\mathrm{mtn}}$ now computes next on the input s^*, by making $r/n+1$ additional queries to the primitive, on the inputs (s^*,i) for $i=0,1,\ldots,r/n$. It forwards to $\mathcal{A}$ the values $(F(s^*,0),F(s^*,1)||\ldots||F(s^*,r/n))$. It proceeds to respond to $\mathcal{A}$'s oracle calls as before and waits for $\mathcal{A}$'s guess bit. It merely forwards the same bit as its guess to the challenger.

  When the challenger's bit is 0, the $\mathcal{A}_{\mathrm{mtn}}$ perfectly simulates towards $\mathcal{A}$ the distribution H_0. When the challenger's bit is 1, $\mathcal{A}$ is given $\mathsf{next}(U_n)$. This corresponds to the hybrid distribution $H_{\frac{1}{2}}$. Thus, the advantage of $\mathcal{A}$ in distinguishing hybrids H_0 and $H_{\frac{1}{2}}$ is upper-bounded by the advantage of $\mathcal{A}_{\mathrm{mtn}}$ in the extraction game which is $\varepsilon^{\mathrm{mtn}}_{\mathsf{MD}}(q+r/n+1,\ell)$.

- Towards bounding the distance between $H_{\frac{1}{2}}$ and H_1, consider the following attacker $\mathcal{A}_{\mathrm{next}}$ against next security of MD: $\mathcal{A}_{\mathrm{next}}$ runs $\mathcal{A}$ answering its oracle queries by passing queries to F. When $\mathcal{A}$ outputs $(x_1,....,x_\ell)$, $\mathcal{A}_{\mathrm{next}}$ returns its own distinguishing challenge, continuing to answer oracle queries for $\mathcal{A}$. In the end, it outputs $\mathcal{A}$'s guess bit.

  It is straight-forward to verify that $\mathcal{A}_{\mathrm{next}}$ perfectly simulates $H_{\frac{1}{2}}$ to $\mathcal{A}$ when given $\mathsf{next}^F(U_n)$ and H_1 when given U_{n+r}. Thus, the advantage of $\mathcal{A}$ in distinguishing hybrids $H_0, H_{\frac{1}{2}}$ is upper-bounded by the advantage of $\mathcal{A}_{\mathrm{next}}$ in the next game which is $\varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q+r/n+1)$.

□

6.5 PRNGs from Merkle-Damgård with Davies-Meyer

This section establishes the robustness of the PRNG construction $\mathsf{DM} = (\mathsf{refresh},\mathsf{next})$ based on Merkle-Damgård with the Davies-Meyer compression function, which is analyzed using an ideal cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. Recall that the construction is defined as follows (cf. Figure 4):

Construction 4 (PRNG from MD-DM)**.** The (n,r)-PRNG construction $\mathsf{DM} = (\mathsf{refresh},\mathsf{next})$ based on Merkle-Damgård with Davies-Meyer (MD-DM) uses a cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and is defined as follows:[19]

[19] The integer arguments to the cipher are to be naturally mapped to $\{0,1\}^n$.

- $\mathsf{refresh}^E(s,x) = E(x,s) \oplus s$, and
- $\mathsf{next}^E(s) = (E(0,s) \oplus s, E(1,s) \oplus s \| \cdots \| E(r/n,s) \oplus s)$.

Note that iteratively absorbing some input blocks $x_1, \ldots, x_\ell$ via $\mathsf{refresh}$, starting with a state s_0 is identical to applying the MD-DM construction to the input with initialization vector (IV) s_0, which is denoted by $\mathsf{MD\text{-}DM}^E_{s_0}(x_1, \ldots, x_\ell)$ in the remainder of this paper.

Theorem 6 (Robustness of MD-DM PRNGs)**.** *Construction 4 is a* $(\gamma^*, q, t, \ell, \varepsilon_{\mathrm{rob}})$*-robust PRNG in the* E*-model for*

$$\varepsilon_{\mathrm{rob}} \;\leq\; 4t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} \right),$$

where $\tilde{q} = q + r/n + 1$.

The robustness of the MD-DM PRNG construction is proved along similar lines as that of the MD construction with a random compression function (cf. Section 6.5). It is highly recommended to read that proof first. The proof again establishes extraction, maintaining, and next security, before showing that these imply recovering and preserving security. The final bound in Theorem 4 follows by applying Theorem 12.

6.5.1 Extraction Security

Lemma 20 (Extraction security)**.** *The advantage of any* γ^**-legitimate* (q,ℓ)*-attacker* $\mathcal{A}$ *against extraction security of* DM *is bounded by*

$$\mathrm{Adv}^{\mathrm{ext},E}_{\mathsf{DM}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{ext}}_{\mathsf{MD}}(\gamma^*, q, \ell) \;:=\; \frac{q^2 + 2(q\ell + \ell^2)}{2^n} + \frac{4q}{2^{\gamma^*}} \;.$$

Proof. The transcript has the identical format as previously, i.e.,

$$\tau \;=\; (s^\star, s_0, x_1, \ldots, x_\ell, L)$$

except that L is now the set of E-queries. To define bad transcripts, let $\ell' \geq 0$ be maximal such that

$$((x_i, y_{i-1}), y_i \oplus y_{i-1}) \in L$$

for some values $y_0, y_1, \ldots, y_{\ell'}$ with $y_0 = s_0$. A transcript τ is a bad transcript if

- $\ell' = \ell$ or
- $\ell' = \ell - 1$ and there exists a query $((x_\ell, *), y_{\ell-1} \oplus s^\star) \in L$.

As before, to apply Theorem 9, one merely needs to bound the probability ratio for good transcripts (Lemma 21) and the probability of a bad transcript occurring in the ideal world, i.e, for $b = 1$ (Lemma 22). □

Lemma 21 (Ratio analysis)**.** *For all* good *transcripts* τ*,*

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \;\geq\; 1 - \frac{2(q\ell + \ell^2)}{2^n} \;.$$

Proof. As in Section 5.1, for a good transcript,

$$\mathsf{p}_1(\tau) \;=\; p_L \cdot 2^{-n} \quad \text{and} \quad \mathsf{p}_0(\tau) \;=\; p_L \cdot q_\tau \;,$$

where p_L denotes the probability that a uniform random cipher is consistent with the queries in L and where q_τ is the probability that

$$\mathsf{MD\text{-}DM}_{s_0}^{E_L}(x_1, x_2, \ldots, x_\ell) \;=\; s^\star$$

over a cipher E_L that is sampled uniformly at random conditioned on being consistent with L.

To derive a lower bound on q_τ, due to τ being a good transcript, it suffices to consider the two cases

(1) $\ell' \leq \ell - 2$ and

(2) $\ell' = \ell - 1$.

For (1), q_τ is the probability (over E_L) that

$$Y_\ell \;:=\; \mathsf{DM}_{y_{\ell'}}^{E_L}(x_{\ell'+1}, \ldots, x_\ell) \;=\; s^\star \;.$$

Consider the intermediate chaining values $Y_{\ell'} = y_{\ell'}, Y_{\ell'+1}, \ldots, Y_{\ell-1}$ (defined via evaluations of E). Moreover, let $L_{\ell'}, L_{\ell'+1}, \ldots, L_{\ell-2}$ be the set of points at which E is defined after evaluating these intermediate values, i.e., $L_{\ell'} := L$ and

$$L_i \;:=\; L_{i-1} \cup \{((x_i, Y_{i-1}), Y_i \oplus Y_{i-1})\}$$

for $i = \ell' + 1, \ldots, \ell - 2$. Furthermore, for a set $\tilde{L}$ of E-queries define the sets $\mathsf{Free\text{-}In}_{\tilde{L}}$ of *free inputs* and $\mathsf{Free\text{-}Out}_{\tilde{L}}$ of *free outputs*, i.e.,

$$y \in \mathsf{Free\text{-}In}_{\tilde{L}} :\Longleftrightarrow ((*, y), *) \notin \tilde{L} \quad \text{and} \quad y \in \mathsf{Free\text{-}Out}_{\tilde{L}} :\Longleftrightarrow ((*, *), y) \notin \tilde{L} \;.$$

Finally, for $i = \ell' + 1, \ldots, \ell - 2$, define the event $\mathsf{FRESH_i}$ that Y_i is a *fresh input*, i.e.,

$$Y_i \in \mathsf{Free\text{-}In}_{L_{i-1}} \;.$$

However, for $Y_{\ell-1}$, let $\mathsf{FRESH}_{\ell-1}$ be the event *not only* that $Y_{\ell-1} \in \mathsf{Free\text{-}In}_{L_{\ell-2}}$ but also that $Y_{\ell-1} \oplus s^\star$ is a free *output*, i.e.,

$$Y_{\ell-1} \oplus s^\star \in \mathsf{Free\text{-}Out}_{L_{\ell-2}} \;.$$

Let,

$$\mathsf{FRESH} \;:=\; \bigcap_{i=\ell'+1}^{\ell-1} \mathsf{FRESH_i} \;.$$

Then, on the one hand,

$$\begin{aligned} \mathsf{P}[Y_\ell = s^\star | \mathsf{FRESH}] \;&=\; \mathsf{P}[E(x_\ell, Y_{\ell-1}) = Y_{\ell-1} \oplus s^\star | \mathsf{FRESH}] \\ &=\; \frac{1}{|\mathsf{Free\text{-}Out}_{L_{\ell-2}}|} \;\geq\; \frac{1}{2^n} \end{aligned}$$

since the conditioning implies that $(x_\ell, Y_{\ell-1})$ is a fresh input to E and that $Y_{\ell-1} \oplus s^\star$ actually is in $\mathsf{Free\text{-}Out}_{L_{\ell-2}}$. On the other hand, in order to bound

$$\mathsf{P}\left[\overline{\mathsf{FRESH}_{\ell-1}} \;\middle|\; \bigcap_{k=\ell'+1}^{\ell-2} \mathsf{FRESH_k}\right] ,$$

observe that, for any values $y_{\ell'+1}, \ldots, y_{\ell-2}$ consistent with the conditioning, $\mathsf{FRESH}_{\ell-1}$ is violated if[20]

$$E_L(x_{\ell-1}, y_{\ell-2}) \in \overline{\mathsf{Free\text{-}In}_{L_{\ell-2}}} \oplus y_{\ell-2}$$

or

$$E_L(x_{\ell-1}, y_{\ell-2}) \in \overline{\mathsf{Free\text{-}Out}_{L_{\ell-2}}} \oplus y_{\ell-2} \oplus s^\star \ .$$

The probability that the former condition is violated is at most

$$\begin{aligned} \frac{|\mathsf{Free\text{-}Out}_{L_{\ell-2}} \cap \overline{\mathsf{Free\text{-}In}_{L_{\ell-2}}} \oplus y_{\ell-2}|}{|\mathsf{Free\text{-}Out}_{L_{\ell-2}}|} &\leq \frac{|\overline{\mathsf{Free\text{-}In}_{L_{\ell-2}}} \oplus y_{\ell-2}|}{|\mathsf{Free\text{-}Out}_{L_{\ell-2}}|} \\ &\leq \frac{q+\ell}{2^n - (q+\ell)} \ . \end{aligned}$$

The same bound via a similar argument is obtained for the latter condition as well as

$$\mathsf{P}\left[\overline{\mathsf{FRESH_i}} \;\middle|\; \bigcap_{k=\ell'+1}^{i-1} \mathsf{FRESH_k}\right] \leq \frac{q+\ell}{2^n - (q+\ell)}$$

for $i = \ell' + 1, \ldots, \ell - 2$. Therefore,

$$\mathsf{P}[\mathsf{FRESH}] \geq 1 - \frac{q\ell + \ell^2}{2^n - (q+\ell)} \ ,$$

and, finally,

$$\begin{aligned} q_\tau &\geq \mathsf{P}[\mathsf{FRESH}] \cdot \mathsf{P}[Y_\ell = s^\star | \mathsf{FRESH}] \\ &\geq \left(1 - \frac{q\ell + \ell^2}{2^n - (q+\ell)}\right) \cdot 2^{-n} \ , \end{aligned}$$

for case (1). For case (2), note that due to $\ell' = \ell - 1$, $(x_\ell, y_{\ell-1})$ is a fresh input to E, and, furthermore, $y_{\ell-1} \oplus s^\star \in \mathsf{Free\text{-}Out}_L$. Thus, in this case the probability that $Y_\ell = s^\star$ is at least 2^{-n}.

Combining both cases, case (1) dominating, one obtains

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \geq 1 - \frac{q\ell + \ell^2}{2^n - (q+\ell)} \geq 1 - \frac{2(q\ell + \ell^2)}{2^n} \ ,$$

using that $q + \ell \leq 2^{n-1}$, an assumption one may always make since the bound in the lemma is vacuous otherwise. □

Lemma 22 (Bad event analysis)**.** *For the set $\mathcal{B}$ of bad transcripts (as defined above),*

$$\mathsf{P}[T_1 \in \mathcal{B}] \leq \frac{4q}{2^{\gamma^*}} + \frac{q^2}{2^n} \ .$$

Proof. The same resampling approach as in the proof of Lemma 15 applies here as well. However, the collisions require additional care. Two queries E queries $((k,u),v)$ and $((k',u'),v')$ are said to *collide* if

$$v \oplus u = v' \oplus u' \ .$$

[20] Using the notation $A \oplus b = \{x \oplus b \mid x \in A\}$.

It is easily verified that the probability, over E, that any two such queries collide is at most $(2^n-1)^{-1}$. Let $\mathcal{E}$ be the event that there exists such a collision in $\mathcal{L}$. Note that

$$\mathsf{P}[T_1 \in \mathcal{B}] \;\leq\; \mathsf{P}[T_1 \in \mathcal{B}|\mathcal{E}] + \mathsf{P}[\overline{\mathcal{E}}] \;\leq\; \mathsf{P}[T_1 \in \mathcal{B}|\mathcal{E}] + \frac{q^2}{2^n} \;.$$

Towards bounding $\mathsf{P}[T_1 \in \mathcal{B}|\mathcal{E}]$, consider a triple $z = (\sigma, L, s_0) \in \mathcal{E}$, which, once more, is shorthand for L being collision-free. Define a *potential chain* as ℓ, for *some* ℓ, values $y_0, y_1, \ldots, y_\ell$ such that $y_0 = s_0$ and, for some values $k_1, \ldots, k_\ell$,

(a) $((k_i, y_{i-1}), y_i \oplus y_{i-1}) \in L$ for $i = 1, \ldots, \ell$ or

(b) $((k_i, y_{i-1}), y_i \oplus y_{i-1}) \in L$ for $i = 1, \ldots, \ell - 1$ and $((k_\ell, y_\ell), y_{\ell-1} \oplus s^\star) \in L$.[21]

Without collisions, L can contain at most $2q$ potential chains; this can be proved by induction: Consider a set collision-free set L' of E-queries and assume that the number of potential chains is at most $2|L'|$. Consider an additional query $((k, u), v)$ that does not cause a collision; it may only (but need not) create a new potential chain in two ways:

- $u = v' \oplus u'$: this corresponds to extending in the sense of (a) above and can only be true for a single previous query $((k', u'), v')$ if L is collision-free;
- $v = v' \oplus u' \oplus s^\star$: this corresponds to creating a chain of type (b) and, again, can only hold for one query $((k', u'), v') \in$ if L is collision-free.

Summarizing, the new query creates at most two new potential chains.

Clearly, conditioned on $Z = z$, $T_1 \in \mathcal{B}$ if and only if for some potential chain, $X_i = k_i$ for all $i = 1, \ldots, \ell$. Hence, by the legitimacy of $\mathcal{A}$,

$$\mathsf{P}[T_1 \in \mathcal{B}|Z = z] \;\leq\; 2q \cdot p_z \;,$$

where $p_z := \mathsf{Pred}(\overline{X}|Z = z)$. The remainder of the proof proceeds in the exact same fashion as the proof of Lemma 15. □

6.5.2 Maintaining Security

The maintaining security of the Davies-Meyer PRNG construction DM is proved along similar lines as that of the MD construction with a random compression function. This section discusses the few differences.

Lemma 23 (Maintaining security)**.** *The advantage of any (q, ℓ)-attacker $\mathcal{A}$ against maintaining security is bounded by*

$$\mathrm{Adv}^{\mathrm{mtn},E}_{\mathsf{DM}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{mtn}}_{\mathsf{DM}}(q, \ell) \;:=\; \frac{2(q\ell + \ell^2)}{2^n} + \frac{q}{2^n} \;.$$

Proof. Similarly to Lemma 16, maintaining security is shown via an H-coefficient proof. Once more, one considers transcripts

$$\tau = (s^\star, s_0, x_1, \ldots, x_\ell, L) \;,$$

with the difference that L refers to E-queries here. A *bad* transcript contains a query of the type $((*, s_0), *) \in L$. Again, the probability of a bad transcript occurring in the $b = 1$ case is at most $|L|/2^n$, and for good transcripts,

$$\mathsf{p}_1(\tau) \;=\; 2^{-2n} \cdot p_L \quad \text{and} \quad \mathsf{p}_0(\tau) \;\geq\; \left(1 - \frac{2(q\ell + \ell^2)}{2^n}\right) \cdot 2^{-2n} \cdot p_L \;,$$

where the latter follows via an argument similar to that in the proof of Lemma 21. □

[21] Observe that in case (b) the "intuitive chain" goes up to $y_{\ell-1}$ but y_ℓ is not part of it.

6.5.3 Next Security

Next security of DM is defined and proved in a fashion analogous to the case with a random compression function.

Lemma 24 (Next security)**.** *The advantage of any q-attacker $\mathcal{A}$ against next security is bounded by*

$$\mathrm{Adv}^{\mathrm{next},E}_{\mathsf{DM}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{next}}_{\mathsf{DM}}(q) \;:=\; \frac{q}{2^n}\ .$$

Proof. For a straight-forward H-coefficient proof, consider the transcript

$$\tau = (s, s^\star_0, s^\star_1, \ldots, s^\star_{r/n}, L)\ ,$$

where s is the initial state, the values $s^\star_i$ are the input to $\mathcal{A}$, and L are the queries $\mathcal{A}$ makes to F. A transcript is *bad* if L contains a query of the form $((i,s),*)$ or $((i,*), s \oplus s^\star_i)$ for some i; otherwise, τ is called *good*. It is easily seen that for good transcripts, the probability ratio is *at least* 1, and, in the ideal world, where $\mathcal{A}$'s view is completely independent of S, the probability of a bad event is at most $q/2^n$. □

6.5.4 Recovering Security

In the following, let $\varepsilon^{\mathrm{ext}}_{\mathsf{DM}}(\gamma^*, q, \ell)$ and $\varepsilon^{\mathrm{next}}_{\mathsf{DM}}(q)$ be as in Lemmas 20 and 24. Once more, extraction and next security together imply recovering security:

Lemma 25 (Recovering Security)**.** *For every γ^*-legitimate (q,ℓ)-attacker $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{rec},E}_{\mathsf{DM}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{ext}}_{\mathsf{DM}}(\gamma^*, \ell, q + r/n + 1) + \varepsilon^{\mathrm{next}}_{\mathsf{DM}}(q + r/n + 1)\ .$$

The proof of the lemma is completely analogous to that of Lemma 18 and is omitted.

6.5.5 Preserving Security

In the following, let $\varepsilon^{\mathrm{mtn}}_{\mathsf{MD}}(q,\ell)$ and $\varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q)$ be as in Lemmas 16 and 17. Maintaining and next security together imply preserving security:

Lemma 26 (Preserving Security)**.** *For every adversary $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{pre},E}_{\mathsf{DM}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{mtn}}_{\mathsf{DM}}(q + r/n + 1, \ell) + \varepsilon^{\mathrm{next}}_{\mathsf{DM}}(q + r/n + 1)\ .$$

The proof of the lemma is completely analogous to that of Lemma 19 and is omitted.

6.6 PRNGs from Sponges

This section establishes the robustness of the PRNG construction $\mathsf{Spg} = (\mathsf{refresh}, \mathsf{next})$ based on the Sponge paradigm with a random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$. Recall that the construction is defined as follows (cf. Figure 5):

Construction 6 (PRNG from Sponges)**.** The Sponge-based PRNG construction $\mathsf{Spg} = (\mathsf{refresh}, \mathsf{next})$ uses a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ to absorb and produce r-bit inputs and outputs, respectively, and is defined as follows:

- $\mathsf{refresh}^\pi(s,x) = \pi(s \oplus x \| 0^c)$, and

- $\mathsf{next}^\pi(s) = (\pi(s) \oplus 0^r \| s^{(\mathsf{c})}, s^{(\mathsf{r})})$.

Note that iteratively absorbing some input blocks $x_1, \ldots, x_\ell$ via $\mathsf{refresh}$, starting with a state s_0 is identical to applying the Sponge construction to the input with initialization vector (IV) s_0, which is denoted by $\mathsf{Sponge}^\pi_{s_0}(x_1, \ldots, x_\ell)$ in the remainder of this section.

Theorem 8 (Robustness of Sponge PRNGs)**.** *Construction 6 is a $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-robust PRNG in the π-model for*

$$\varepsilon_{\text{rob}} \;\leq\; 4t \cdot \left(\frac{\tilde{q}^2 + \tilde{q}\ell + \ell^2}{2^n} + \frac{\tilde{q}}{2^{\gamma^*}} + \frac{\tilde{q}^2}{2^c} \right),$$

where $\tilde{q} = q + r/n + 1$.

The proof of the robustness of the Sponge PRNG construction follows the same outline as that of the MD construction (cf. Section 6.4). It is highly recommended to read that proof first. One crucial difference between the Merkle-Damgård constructions and Sponges is that Sponges do *not* satisfy extraction/maintaining security (with good parameters). For example, given the state of a Sponge PRNG after absorbing a single (possibly high-entropy) input, a simple inverse query to π results in a value of the form $a\|0^c$, which is unlikely to happen in the ideal world ($b = 1$). This is handled by explicitly introducing a "hit" probability, i.e., the probability that the attacker queries π^{-1} on the final state of the Sponge. Recovering and preserving security are then established by arguing that the hit probability is low when next is applied to the state. The final bound in Theorem 8 follows by applying Theorem 12.

6.6.1 Extraction Security

The extraction security of the Sponge PRNG construction is defined and proved along similar lines as that of the previous constructions. This section discusses the differences. Denote by $\mathrm{Adv}^{\text{inv},\pi}_{\mathsf{Spg}}(\mathcal{A})$ the probability that $\mathcal{A}$ queries π^{-1} on the value $s^\star$ returned by the challenger.

Lemma 27 (Extraction security)**.** *The advantage of any γ^*-legitimate (q, ℓ)-attacker $\mathcal{A}$ against extraction security of* DM *is bounded by*

$$\mathrm{Adv}^{\text{ext},\pi}_{\mathsf{Spg}}(\mathcal{A}) \;\leq\; \varepsilon^{\text{ext}}_{\mathsf{Spg}}(\gamma^*, \ell, q) \;:=\; \frac{q + 2(q\ell + \ell^2)}{2^n} + \frac{2q}{2^{\gamma^*}} + \frac{q^2}{2^c} + \mathrm{Adv}^{\text{inv},\pi}_{\mathsf{Spg}}(\mathcal{A}) \;.$$

In the following, for convenience, let

$$\delta^{\text{ext}}_{\mathsf{Spg}}(\gamma^*, \ell, q) \;:=\; \frac{q + 2(q\ell + \ell^2)}{2^n} + \frac{2q}{2^{\gamma^*}} + \frac{q^2}{2^c} \;.$$

Proof. The transcript has the identical format as previously, i.e.,

$$\tau \;=\; (s^\star, s_0, x_1, \ldots, x_\ell, L)$$

except that L is now the set of π-queries. To define bad transcripts let $\ell' \geq 0$ be maximal such that there exist $(u_i, v_i) \in L$ with $u_1 = s_0 \oplus x_1\|0^c$ and

$$u_i = v_{i-1} \oplus x_i\|0^c$$

for $i = 2, \ldots, \ell - 1$. A transcript τ is a bad transcript if

- **(hit)** $(*, s^\star) \in L$ or

- **(chain)** $\ell' = \ell$.

As before, to apply Theorem 9, one merely needs to bound the probability ratio for good transcripts (Lemma 28) and the probability of a bad transcript occurring in the ideal world, i.e, for $b = 1$ (Lemma 29). □

Lemma 28 (Ratio analysis)**.** *For all* good *transcripts* τ*,*

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \;\geq\; \left(1 - \frac{2(q\ell + \ell^2)}{2^n}\right) .$$

Proof. As in Section 5.1, for a good transcript,

$$\mathsf{p}_1(\tau) \;=\; p_L \cdot 2^{-n} \quad \text{and} \quad \mathsf{p}_0(\tau) \;=\; p_L \cdot q_\tau \;,$$

where p_L denotes the probability that a uniform random permutation is consistent with the queries in L and where q_τ is the probability that

$$\mathsf{Sponge}^{\pi_L}_{s_0}(x_1, x_2, \ldots, x_\ell) \;=\; s^\star$$

over a permutation π_L that is sampled uniformly at random conditioned on being consistent with L.

To derive a lower bound on q_τ, note that due to τ being a good transcript $\ell' < \ell$. Hence, q_τ is the probability (over π_L) that

$$V_\ell \;:=\; \mathsf{Sponge}^{\pi_L}_{v_{\ell'}}(x_{\ell'+1}, \ldots, x_\ell) \;=\; s^\star \;.$$

Consider the intermediate values $U_{\ell'+1}, V_{\ell'+1}, U_{\ell'+2}, V_{\ell'+2}, \ldots, V_{\ell-1}, U_\ell$, where

$$U_{\ell'+1} \;=\; v_{\ell'} \oplus x_{\ell'+1} \| 0^c$$

and

$$U_i = V_{i-1} \oplus x_i \| 0^c$$

for $i = \ell' + 2, \ldots, \ell$ as well as

$$V_i = \pi(U_i)$$

for $i = \ell' + 1, \ldots, \ell - 1$. Define, for $i = \ell' + 1, \ldots, \ell$, the event $\mathsf{FRESH_i}$ that

- U_i is *fresh*, i.e., there is no query of type $(U_i, *) \in L$ and $U_i \neq U_j$ for $j < i$, and
- V_{i-1} is not a *hit*, i.e., $V_i \neq s^\star$;

observe that $\mathsf{FRESH}_{\ell'+1}$ is always true due to the maximality of ℓ' and the fact that τ is a good transcript. Let,

$$\mathsf{FRESH} \;:=\; \bigcap_{i=\ell'+1}^{\ell} \mathsf{FRESH_i} \;.$$

Then,

$$\mathsf{P}[V_\ell = s^\star | \mathsf{FRESH}] \;\geq\; 2^{-n}$$

since the conditioning implies that U_ℓ is a fresh input and therefore V_ℓ is chosen uniformly form a set of size at most 2^n, which contains $s^\star$ due to τ being a good transcript and no hits occurring while evaluating the intermediate values.

Moreover, observe that if U_{i-1} is a fresh input, the probability that V_{i-1} hits is at most $(2^n - (q+\ell))^{-1}$, and the probability that U_i is not fresh is at most $(q+\ell)(2^n-(q-\ell))^{-1}$ as there are at most $q+\ell$ non-fresh values when U_i is sampled uniformly from a set of size at least $2^n-(q+\ell)$. Hence,

$$\mathsf{P}\left[\overline{\mathsf{FRESH_i}} \;\middle|\; \bigcap_{k=\ell'+1}^{i-1} \mathsf{FRESH_k}\right] \;\leq\; \frac{q+\ell+1}{2^n-(q+\ell)},$$

and

$$\mathsf{P}[\mathsf{FRESH}] \;\geq\; 1 - \frac{q\ell+\ell^2+\ell}{2^n-(q-\ell)} \;,$$

and, finally,

$$\begin{aligned} q_\tau \;&\geq\; \mathsf{P}[\mathsf{FRESH}] \cdot \mathsf{P}[Y_\ell = s^\star | \mathsf{FRESH}] \\ &\geq\; \left(1 - \frac{q\ell+\ell^2+\ell}{2^n-(q-\ell)}\right) \cdot 2^{-n} \;, \end{aligned}$$

which implies

$$\frac{\mathsf{p}_0(\tau)}{\mathsf{p}_1(\tau)} \;\geq\; 1 - \frac{q\ell+\ell^2}{2^n-(q-\ell)} \;\geq\; 1 - \frac{2(q\ell+\ell^2)}{2^n} \;.$$

□

Lemma 29 (Bad event analysis)**.** *For the set $\mathcal{B}$ of bad transcripts (as defined above),*

$$\mathsf{P}[T_1 \in \mathcal{B}] \;\leq\; \frac{2q}{2^{\gamma^*}} + \frac{q}{2^n} + \frac{q^2}{2^c} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathsf{Spg}}(\mathcal{A}) \;.$$

Proof. The same resampling approach as in the proof of Lemma 15 applies here as well. First, observe that a hit occurs if either one of the forward queries returns $s^\star$ or if the attacker makes a backward query on $s^\star$. Hence, the probability of a hit is at most

$$\frac{q}{2^n} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathsf{Spg}}(\mathcal{A}) \;.$$

Consider the following directed graph $G = (V, E)$ based on the query set L:

- the nodes are the capacity parts that appear in L, i.e.,

$$V \;=\; \{v^{(\mathsf{c})} \mid (v,*) \in L \vee (*,v) \in L\} \;;$$

- two nodes are connected by a labeled edge if a corresponding query has been made, i.e.,

$$E \;=\; \{(u^{(\mathsf{c})}, v^{(\mathsf{c})}, l) \mid (u,v) \in L \;\wedge\; l = u^{(\mathsf{r})} \oplus v^{(\mathsf{r})}\} \;.$$

L is called *collision-free* if there is at most one path *with unique labels* from $s_0^{(\mathsf{c})}$ to every other node in G. Let $\mathcal{E}$ be the event that L is *not* collision-free. Note that

$$\mathsf{P}[T_1 \in \mathcal{B}] \;\leq\; \mathsf{P}[T_1 \in \mathcal{B} | \mathcal{E}] + \mathsf{P}[\overline{\mathcal{E}}] \;\leq\; \mathsf{P}[T_1 \in \mathcal{B} | \mathcal{E}] + \frac{q^2}{2^c} \;.$$

Towards bounding $\mathsf{P}[T_1 \in \mathcal{B}|\mathcal{E}]$, consider a triple $z = (\sigma, L, s_0) \in \mathcal{E}$, which, once more, is shorthand for L being collision-free. Define a *potential chain* to be, for *some* ℓ, any sequence $(u_1, v_1), \ldots, (u_\ell, v_\ell) \in L$ such that $u_1 = s_0 \oplus \mu_1 \| 0^c$ and

$$u_i = v_{i-1} \oplus \mu_i \| 0^c$$

for $i = 2, \ldots, \ell - 1$ and some values $\mu_1, \ldots, \mu_\ell$. Note that $(u_1^{(\mathsf{c})}, v_1^{(\mathsf{c})}), \ldots, (u_\ell^{(\mathsf{c})}, v_\ell^{(\mathsf{c})})$ describe a path from $s_0^{(\mathsf{c})}$ to $v_\ell^{(\mathsf{c})}$ with labels $\mu_1, \ldots, \mu_\ell$. Hence, that for a collision-free L, there are at most q potential chains.

Clearly, conditioned on $Z = z$, $T_1 \in \mathcal{B}$ if and only if for some potential chain, $X_i = \mu_i$ for all $i = 1, \ldots, \ell$. Hence, by the legitimacy of $\mathcal{A}$,

$$\mathsf{P}[T_1 \in \mathcal{B} | Z = z] \ \leq \ q \cdot p_z \ ,$$

where $p_z := \mathsf{Pred}(\overline{X} | Z = z)$. The remainder of the proof proceeds in the exact same fashion as the proof of Lemma 15. □

6.6.2 Maintaining Security

The maintaining security of the Sponge PRNG construction Spg is proved along similar lines as that of the previous constructions. This section discusses the differences.

Lemma 30 (Maintaining security)**.** *The advantage of any (q, ℓ)-attacker $\mathcal{A}$ against maintaining security is bounded by*

$$\mathrm{Adv}^{\mathrm{mtn},\pi}_{\mathsf{Spg}}(\mathcal{A}) \ \leq \ \varepsilon^{\mathrm{mtn}}_{\mathsf{Spg}}(q, \ell) \ := \ \frac{2(q\ell + \ell^2)}{2^n} + \frac{2q}{2^n} + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathsf{Spg}}(\mathcal{A}) \ .$$

In the following, for convenience, let

$$\delta^{\mathrm{mtn}}_{\mathsf{Spg}}(\gamma^*, \ell, q) \ := \ \frac{2(q\ell + \ell^2)}{2^n} + \frac{2q}{2^n} \ .$$

Proof. Similarly to the preceding maintaining proofs, maintaining security of Spg is shown via an H-coefficient proof. Once more, one considers transcripts

$$\tau = (s^\star, s_0, x_1, \ldots, x_\ell, L) \ ,$$

where L refers to π-queries here. In a *bad* transcript, L contains a query of the type

- $(s_0, *)$, which happens (with in the case $b = 1$) with probability at most $|L|/2^n$ since s_0 is completely independent of $\mathcal{A}$'s view, or
- $(*, s^\star)$, which happens (in the case $b = 1$) with probability $|L|/2^n$ via a forward query or with probability $\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathsf{Spg}}(\mathcal{A})$ via a backward query.

As for good transcripts

$$\mathsf{p}_1(\tau) \ = \ 2^{-2n} \cdot p_L \quad \text{and} \quad \mathsf{p}_0(\tau) \ \geq \ \left(1 - \frac{2(q\ell + \ell^2)}{2^n}\right) \cdot 2^{-2n} \cdot p_L \ ,$$

where the latter follows via an argument similar to that in the proof of Lemma 28. □

6.6.3 Next Security

Recall that the next function next of the Sponge construction computes, on an input state s_0,

$$(s, y) \ = \ \mathsf{next}^\pi(s_0) \ = \ (\pi(s_0) \oplus 0^r \| s_0^{(\mathsf{c})}, s_0^{(\mathsf{r})}) \ ,$$

where s is the new state and y is the output. Next security demands that if s_0 is chosen uniformly at random, then the output of next be indistinguishable from U_{n+r} to an attacker $\mathcal{A}$ making at most to q queries to π. Denote by $\mathrm{Adv}^{\mathrm{next},\pi}_{\mathsf{Spg}}(\mathcal{A})$ the advantage of $\mathcal{A}$.

Lemma 31 (Next security)**.** *The advantage of any q-attacker $\mathcal{A}$ against next security is bounded by*

$$\mathrm{Adv}^{\mathrm{next},\pi}_{\mathsf{Spg}}(\mathcal{A}) \;\leq\; \varepsilon^{\mathrm{next}}_{\mathsf{Spg}}(q) \;:=\; \frac{2q}{2^c}\ .$$

Proof. For a simple H-coefficient proof, consider a transcript

$$\tau = (s_0^{(\mathsf{c})}, s^\star, y^\star, L),$$

where s_0 is the initial state, $s^\star$ is the new state, $y^\star$ is the output value, and L are the queries to π. A bad transcript is a transcript with a query of the type

- $(y^\star \| s_0^{(\mathsf{c})}, *) \in L$ or
- $(*, s^\star \oplus 0^r \| s_0^{(\mathsf{c})}) \in L$.

Since the view of $\mathcal{A}$ in the ideal world, where $\mathcal{A}$'s view is independent of $s_0^{(\mathsf{c})}$, the probability of a bad transcript is at most $2q/2^c$.

For a good transcript, observe that

$$\mathsf{p}_1(\tau) \;=\; 2^{-c} \cdot 2^{-r} \cdot 2^{-n} \cdot p_L\ ,$$

where p_L denotes the probability that a uniform random permutation is consistent with the queries in L. Moreover,

$$\mathsf{p}_0(\tau) \;=\; 2^{-c} \cdot 2^{-r} \cdot q_\tau \cdot p_L\ .$$

Note that since τ is a good transcript, $(y^\star, s_0^{(\mathsf{c})})$ is a fresh input to π and $s \oplus 0^r \| s_0^{(\mathsf{c})}$ is still available; hence $q_\tau \geq 2^{-n}$. □

6.6.4 Recovering Security

In the following, let $\varepsilon^{\mathrm{ext}}_{\mathsf{Spg}}(\gamma^*, q, \ell)$ and $\varepsilon^{\mathrm{next}}_{\mathsf{Spg}}(q)$ be as in Lemmas 27 and 31. Once more, extraction and next security together imply recovering security:

Lemma 32 (Recovering Security)**.** *For every γ^*-legitimate (q,ℓ)-attacker $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{rec},\pi}_{\mathsf{Spg}}(\mathcal{A}) \;\leq\; \delta^{\mathrm{ext}}_{\mathsf{Spg}}(\gamma^*, \ell, q + r/n + 1) + \frac{q + r/n + 1}{2^n} + 2 \cdot \varepsilon^{\mathrm{next}}_{\mathsf{Spg}}(q + r/n + 1)\ .$$

Proof. As in the proof of Lemma 18 consider, for $b \in \{0,1\}$, the recovering experiment H_b conditioned on the secret bit having the value b. Moreover, again define a hybrid experiment $H_{\frac{1}{2}}$ in which the challenge oracle returns $\mathsf{next}^\pi(U_n)$ to $\mathcal{A}$.

- The distance between H_0 and $H_{\frac{1}{2}}$ is bounded by a similar reduction $\mathcal{A}_{\mathrm{ext}}$ to extraction security as in Lemma 18. Hence, it is at most

  $$\varepsilon^{\mathrm{ext}}_{\mathsf{Spg}}(\gamma^*, \ell, q + r/n + 1) \;\leq\; \delta^{\mathrm{ext}}_{\mathsf{Spg}}(\gamma^*, \ell, q + r/n + 1) + \mathrm{Adv}^{\mathrm{inv},\pi}_{\mathsf{Spg}}(\mathcal{A}_{\mathrm{ext}})\ .$$

  In order to bound the probability of a hit in the ideal world of extraction security, one analyses this event in a hybrid world similar to H_1. By next security and the fact that hit occurs with probability at most $(q + r/n + 1) \cdot 2^{-n}$ in H_1 in that hybrid,

  $$\mathrm{Adv}^{\mathrm{inv},\pi}_{\mathsf{Spg}}(\mathcal{A}_{\mathrm{ext}}) \;\leq\; \frac{q + r/n + 1}{2^n} + \varepsilon^{\mathrm{next}}_{\mathsf{Spg}}(q + r/n + 1)\ .$$

- By an argument similar to that in Lemma 18, one uses next security to show that the advantage of $\mathcal{A}$ in distinguishing hybrids $H_{\frac{1}{2}}$ and H_1 is upper-bounded by $\varepsilon^{\mathrm{next}}_{\mathsf{MD}}(q)$.

□

6.6.5 Preserving Security

In the following, let $\varepsilon^{\mathrm{mtn}}_{\mathsf{Spg}}(q,\ell)$ and $\varepsilon^{\mathrm{next}}_{\mathsf{Spg}}(q)$ be as in Lemmas 27 and 31. Once more, maintaining and next security together imply preserving security:

Lemma 33 (Preserving Security)**.** *For every (q,ℓ)-attacker $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{pre},\pi}_{\mathsf{Spg}}(\mathcal{A}) \;\le\; \delta^{\mathrm{mtn}}_{\mathsf{Spg}}(\ell, q + r/n + 1) + \frac{q + r/n + 1}{2^n} + 2 \cdot \varepsilon^{\mathrm{next}}_{\mathsf{Spg}}(q + r/n + 1) \;.$$

The proof proceeds similarly to that of Lemma 32—except that reductions to maintaining security are made instead of extraction security—and is omitted.

7 Security Proofs for IT Constructions

This section analyzes all previously presented IT secure constructions. Used by all proofs in this section are the three propositions stated and proved in Section 7.1. Section 7.2 then shows the security of the monolithic extractor from Section 3.2. While rather straight-forward, the corresponding proof is quite instructive for the security proofs of the PRNG constructions from Section 5. For these proofs, it is convenient to define and establish an intermediate property, which is discussed in Section 7.3. Finally, Sections 7.4 and 7.5 establish the robustness (cf. Section 4.2) of the PRNG constructions.

7.1 Information-Theoretic Preliminaries

The *collision probability* of a random variable X is defined simply as $\mathsf{Coll}(X) := \sum_x \mathsf{P}[X = x]^2$. Moreover, let $\mathsf{Coll}(X|y) := \sum_x \mathsf{P}[X = x|Y = y]^2$, and define the conditional collision probability

$$\mathsf{Coll}(X|Y) \;:=\; \mathbf{E}_{y \leftarrow Y}\big[\mathsf{Coll}(X|y)\big] \;.$$

The following two propositions relate the statistical distance from uniform to the collision probability. The first is well known and at the core of the proof of the leftover hash lemma [29]; the latter is proved for self-containment.

Proposition 1. *For any random variable X with size-N range, and a uniformly distributed U with the same range,*

$$\mathsf{SD}(X,U) \;\le\; \frac{1}{2}\sqrt{N \cdot \mathsf{Coll}(X) - 1} \;.$$

Proposition 2. *Let F be chosen uniformly at random from a set $\mathcal{F}$. Then, for any random variable X with size-N range (arbitrarily correlated with $\mathcal{F}$), and a uniformly distributed U with the same range, independent of F,*

$$\mathsf{SD}((X,F),(U,F)) \;\le\; \frac{1}{2}\sqrt{N \cdot \mathsf{Coll}(X|F) - 1} \;.$$

Proof. By Proposition 1,

$$\mathsf{SD}((X,F),(U,F)) \;\le\; \frac{1}{2}\sqrt{N|\mathcal{F}| \cdot \mathsf{Coll}(X,F) - 1} \;.$$

Moreover,

$$\begin{aligned}\mathsf{Coll}(X,F) &= \sum_{x,f}\mathsf{P}[(X,F)=(y,f)]^2\\ &= \frac{1}{|\mathcal{F}|}\sum_{f}\mathsf{P}[F=f]\sum_{x}\mathsf{P}[X=x|F=f]^2\\ &= \frac{\mathsf{Coll}(X|F)}{|\mathcal{F}|}\ ,\end{aligned}$$

from which the proposition follows. □

The following proposition will also be useful.

Proposition 3. *Consider two random variables X and Y with identical range and let $\mathcal{E}$ and $\mathcal{E}'$ be events on their respective probability spaces. Assume $\mathsf{P}[\mathcal{E}]=\mathsf{P}[\mathcal{E}']$, then*

$$\mathsf{SD}(X,Y) \ \leq\ \mathsf{SD}(X|\mathcal{E},Y|\mathcal{E}')+\mathsf{P}[\overline{\mathcal{E}}]\ .$$

Proof. Observe that

$$\begin{aligned}\mathsf{SD}(X,Y) &= \frac{1}{2}\sum_{x}|P_X(x)-P_Y(y)|\\ &= \frac{1}{2}\sum_{x}\left|P_{X|\mathcal{E}}(x)\mathsf{P}[\mathcal{E}]+P_{X|\overline{\mathcal{E}}}(x)\mathsf{P}[\overline{\mathcal{E}}]-P_{Y|\mathcal{E}'}(x)\mathsf{P}[\mathcal{E}']-P_{Y|\overline{\mathcal{E}'}}(x)\mathsf{P}[\overline{\mathcal{E}'}]\right|\\ &\leq \mathsf{P}[\mathcal{E}]\cdot\frac{1}{2}\sum_{x}\left|P_{X|\mathcal{E}}(x)-P_{Y|\mathcal{E}'}(x)\right|+\mathsf{P}[\overline{\mathcal{E}}]\cdot\frac{1}{2}\sum_{x}\left|P_{X|\overline{\mathcal{E}}}(x)-P_{Y|\overline{\mathcal{E}'}}(x)\right|\\ &\leq \mathsf{SD}(X|\mathcal{E},Y|\mathcal{E}')+\mathsf{P}[\overline{\mathcal{E}}]\ .\end{aligned}$$

□

7.2 Monolithic Extractor

From Section 3.2, recall the monolithic extractor mono defined to work with a random oracle $F:\{0,1\}^m\to\{0,1\}^n$:

Construction 1 (Monolithic extractor)**.** The monolithic seedless extractor $\mathsf{mono}^G:\{0,1\}^m\to\{0,1\}^n$ using a random oracle $G:\{0,1\}^m\to\{0,1\}^n$ is defined by

$$\mathsf{mono}^G(x) := G(x)\ .$$

Construction 1 is an IT seedless extractor according to Definition 3.

Theorem 2 (Monolithic seedless IT-extraction)**.** *Construction* mono *is a (γ^*,q,ε)*-IT-extractor in the G-model *for*

$$\varepsilon\ \leq\ \frac{1}{2}\sqrt{\frac{2^{-(\gamma^*-n)}}{1-\rho}}+\rho\ ,$$

where $\rho=q/2^{\gamma^*}$.

The proof of Theorem 2 proceeds by bounding the statistical distance of $\mathcal{A}_2$'s views in the real and ideal experiments via the corresponding collision probabilities (as done in the proof of the left-over hash lemma). In the proofs of the actual PRNG constructions in the following sections, bounding said collision probabilities constitutes the bulk of the proof and is quite involved.

Proof. Consider the extraction game corresponding to Definition 3 (cf. Section 3.1). Since $\mathcal{A}_2$ gets to make an unbounded number of queries to the random oracle G, one may equivalently consider the game where $\mathcal{A}_2$ simply gets the entire function table of G as input. Therefore, the distinguishing advantage of $\mathcal{A}_2$ is upper bounded by

$$\mathsf{SD}((\Sigma, Y_0, \mathcal{L}_1, G), (\Sigma, Y_1, \mathcal{L}_1, G)) \ ,$$

where Σ is (the random variable corresponding to) the state information output by $\mathcal{A}_1$, $Y_0 = G(X)$, Y_1 a uniform random string, and $\mathcal{L}_1$ the query/answer list by $\mathcal{A}_1$.

Let $\mathcal{E}$ be the event that $\mathcal{A}_1$ does not query the value X it provides as input to the extractor. Observe that this event can be defined in both the real ($b = 0$) and ideal ($b = 1$) experiments. In the ideal experiment, the probability of $\mathcal{E}$ not occurring is easy to bound: Since Y_1 is uniformly random and independent of X, consider the following equivalent way of sampling a tuple $(\Sigma, X, Y_1, \mathcal{L}_1, G)$:

1. Sample Y_1 and G uniformly at random.
2. Run $\mathcal{A}_1^G$ to produce Σ, $\tilde{X}$, and $\mathcal{L}_1$.
3. Rerun $\mathcal{A}_1^G$ with fresh randomness, but conditioned on the state information being Σ and the queries being $\mathcal{L}_1$. This results in a new value X.
4. Output $(\Sigma, X, Y_1, \mathcal{L}_1, G)$.

It is easily seen that the distribution produced via resampling is identical to that of the actual experiment.

Note that for particular values σ and L_1,

$$\mathsf{P}[\overline{\mathcal{E}}|\sigma L_1] \ \leq \ q \cdot \mathsf{Pred}(X|\sigma L_1) \ ,$$

which can be easily seen due to the alternative sampling above. Hence, taking expectations,

$$\mathsf{P}[\overline{\mathcal{E}}] \ \leq \ q \cdot \mathsf{Pred}(X|\Sigma\mathcal{L}_1) \ \leq \ q \cdot 2^{-\gamma^*} \ ,$$

using the γ^*-legitimacy of $\mathcal{A}$ in the last step.

Using Proposition 3, one obtains

$$\mathsf{SD}((\Sigma, Y_0, \mathcal{L}_1, G), (\Sigma, Y_1, \mathcal{L}_1, G)) \ \leq \ \mathsf{SD}((\Sigma, Y_0, \mathcal{L}_1, G)|\mathcal{E}, (\Sigma, Y_1, \mathcal{L}_1, G)|\mathcal{E}) + q \cdot 2^{-\gamma^*} \ .$$

In order to bound the statistical distance conditioned on $\mathcal{E}$, condition additionally on arbitrary values $z = (\sigma, L_1)$ (with non-zero probability given $\mathcal{E}$). Observe that under such conditioning, G is chosen uniformly at random from all functions consistent with G. Using Proposition 2, one bounds the desired statistical distance as

$$\mathsf{SD}((Y_0, G)|z\mathcal{E}, (Y_1, G)|z\mathcal{E}) \ \leq \ \frac{1}{2}\sqrt{2^n \cdot \mathsf{Coll}(Y_0|Gz\mathcal{E}) - 1} \ .$$

To bound the collision probability $\mathsf{Coll}(Y|Gz\mathcal{E})$, consider the following experiment:[22]

[22] Observe that, in general, $\mathsf{Coll}(U|V)$ is equal to the probability that $U = U'$ in the experiment where one jointly samples U and V, and then resamples U' conditioned on the value of V.

1. Sample G uniformly at random consistent with L_1.
2. Sample inputs $X \leftarrow \mathcal{A}_1^G$ and $X' \leftarrow \mathcal{A}_1^G$ independently but conditioned on $Z = z$ and $\mathcal{E}$.
3. Compute $Y_0 \leftarrow \mathsf{mono}^G(X)$ and $Y_0' \leftarrow \mathsf{mono}^G(X')$ respectively.

The collision probability $\mathsf{Coll}(Y|Gz)$ is therefore equal to

$$\mathsf{P}[Y_0 = Y_0'] \;\leq\; \mathsf{P}[X = X'] + \mathsf{P}[Y = Y'|X \neq X']$$

in the experiment above. The former term is at most $p_z := \mathsf{Pred}(X|Z = z, \mathcal{E})$. For the latter term, consider arbitrary $x \neq x'$. Since neither x nor x' is covered by L_1, $Y_0 = Y_0'$ occurs with probability 2^{-n}. Hence,

$$\mathsf{SD}((Y_0, G)|z\mathcal{E}, (Y_1, G)|z\mathcal{E}) \;\leq\; \frac{1}{2}\sqrt{2^n(p_z + 2^{-n}) - 1} \;\leq\; \frac{1}{2}\sqrt{2^n p_z}\ .$$

Using Jensen's inequality, one obtains

$$\begin{aligned}\mathsf{SD}((Y_0, G)|z\mathcal{E}, (Y_1, G)|z\mathcal{E}) \;&\leq\; \frac{1}{2}\sqrt{2^n \cdot \mathsf{Pred}(X|Z\mathcal{E})} \\ &\leq\; \frac{1}{2}\sqrt{\frac{2^{-(\gamma^* - n)}}{(1-\rho)}}\ ,\end{aligned}$$

where the last inequality follows from

$$\begin{aligned}\mathsf{Pred}(X|Z) \;&\geq\; \sum_z \mathsf{P}[Z = z \wedge \mathcal{E}] \cdot p_z \\ &=\; \mathsf{P}[\mathcal{E}] \cdot \sum_{z \in \mathcal{E}} \frac{\mathsf{P}[Z = z \wedge \mathcal{E}]}{\mathsf{P}[\mathcal{E}]} \cdot p_z \\ &=\; \mathsf{P}[\mathcal{E}] \cdot \mathsf{Pred}(\overline{X}|Z\mathcal{E})\ .\end{aligned}$$

and $\mathsf{Pred}(X|Z) \leq 2^{-\gamma^*}$, by the legitimacy of $\mathcal{A}$. □

7.3 Intermediate IT PRNG Security Notions

Similarly to the computational case, it is useful to consider a simplified security notion, *recovering* security, for IT PRNGs. Recall that for the computational case, recovering security requires that the output of the next function next look random once sufficient amounts of entropy have been accumulated in the PRNG's state. In the IT case, the requirement is relaxed by only requiring that the output be indistinguishable from $(0^n, U_r)$. That is, call to next always resets the PRNG state to 0^n. This is without loss, as the definition of legitimacy considers every call to next an entropy drain.

The IT recovering game is represented in Figure 9. The advantage of an attacker $\mathcal{A}$ in this game is denoted by $\mathrm{Adv}^{\text{rec-IT},P}_{\mathsf{PRNG}}(\mathcal{A})$. An attacker $\mathcal{A}$ in this game is γ^*-legitimate if

$$\mathrm{H}_\infty(X_1, \ldots, X_\ell|\Sigma\mathcal{L}S_0) \;\geq\; \gamma^*\ ,$$

where

- Σ is the state of $\mathcal{A}$ just before the call to **chall**,

Oracles of PRNG Recovering-IT Game

init
  $b \leftarrow \{0,1\}$

chall $(s_0, x_1, \ldots, x_\ell)$
  for $i = 1, \ldots, \ell$
    $s_i \leftarrow$ $\mathsf{refresh}(s_{i-1}, x_i)$
  if $b = 0$
    return $\mathsf{next}(s_\ell)$
  else
    $y \leftarrow \{0,1\}^r$
    return $(0^n, y)$

Figure 9: *Oracles for PRNG Recovering-IT game.*

- $\mathcal{L}$ is the list of query and answers $\mathcal{A}$ has made to P up to the call to **chall**, and
- S_0 is the initial state that the adversary provides.

A (q, ℓ)-IT-attacker here is one that can make at most q queries to P before its challenge (and arbitrarily many afterwards), and such that the input to **chall** consists of at most ℓ blocks.

Definition 13. *A PRNG construction* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *is said to be* $(\gamma^*, q, \ell, \varepsilon)$-IT-recovering *in the P-model if for every γ^*-IT-legitimate (q, ℓ)-IT-attacker,*

$$\mathrm{Adv}^{\text{rec-IT},P}_{\mathsf{PRNG}}(\mathcal{A}) \;\leq\; \varepsilon \;.$$

Observe that the definition of recovering security for IT PRNGs is—up to syntactical differences—equivalent to the security of the PRNG as an extractor, which is obtained by replacing the PRNG next algorithm by an algorithm $\mathsf{finalize}$ that simply discards the state output by next. In other words, an IT-robust PRNG can be viewed as an IT-secure online extractor. In particular, the following theorem is in principle little more than just a union bound.

Theorem 34. *Let* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *be a PRNG for which* $\mathsf{refresh}$ *makes α P-calls and* next *makes β P-calls. Let the* PRNG *be $(\gamma^*, q, \ell, \varepsilon_{\text{rec}})$-IT-recovering in the P-model. Then,* PRNG *is also $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-IT-robust in the P-model, where*

$$\varepsilon_{\text{rob}} \;\leq\; t \cdot \varepsilon_{\text{rec}} \;.$$

The proof can be found in Appendix C.2.

7.4 IT PRNGs from Merkle-Damgård

This section establishes the robustness of the PRNG construction $\mathsf{MD}_r = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with a random compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. Recall that the construction is defined as follows (cf. Figure 3):

Construction 3 (IT-PRNG from Merkle-Damgård)**.** The (m, r)-PRNG construction $\mathsf{MD}_r = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is defined as follows:

- $\mathsf{refresh}^F(s, x) = F(s, x)$, and
- $\mathsf{next}^F(s) = (0^n, s[1..r])$.

Construction 3 achieves the following security:

Theorem 5 (IT-Robustness of Merkle-Damgård PRNGs)**.** *Construction 3 is a* $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$*-IT-robust PRNG in the F-model, where*

$$\varepsilon_{\text{rob-it}} \;\leq\; \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 \cdot \frac{\tilde{q}^2 2^r}{2^{2n}}} \;+\; t\rho \;,$$

for $\rho = \frac{\tilde{q}^2}{2^r}$ *where* $\tilde{q} = q + t\ell$*.*

The theorem will be a direct consequence of the following lemma, and Theorem 34 above, which adds a multiplicative factor t to obtain the bound in the theorem.

Lemma 35. *For every* γ^**-IT-legitimate* (q, ℓ)*-attacker in the ideal compression function model,*

$$\mathrm{Adv}^{\text{rec-IT},\gamma^*}_{\mathsf{MD}_r}(\mathcal{A}) \;\leq\; \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 \frac{q^2 2^r}{2^{2n}}} + \rho \quad ,$$

where $\rho = \frac{q^2}{2^r}$*.*

Before we turn to a proof, we note the challenges here. In particular, the core of the proof will be to show a bound on the collision probability of the output of next for a random compression function F, which in turn will reduce to outputting the truncation of the output of the MD construction. This problem resembles that studied by [17] and by [24]. However, the difficulty here is that we need to consider the set of queries previously done by the adversary $\mathcal{A}$. This will significantly complicate the proof—we also modify slightly the graph-theoretic formalization adopted by these previous works to something more amenable to our more complex setting.

Proof (of Lemma 35). We define a few random variables which we will be using in our proofs.

- F a randomly chosen compression function, to which the adversary is given access. We use F both for the oracle itself, as well as for the random variable describing the entire function table.
- ℓ: Number of blocks input to the challenge oracle (which is a random variable itself, we overload notation here, using the same letter we use in the bound on ℓ in the lemma statement).
- $\overline{X} = (\overline{X}_1, \ldots, \overline{X}_\ell)$: the blocks input to the challenge oracle.
- $\tilde{Y}_\ell$: output of MD_r. Let us remind ourselves that this is s truncated to r bits (the output of next);
- $Z = (\Sigma, \mathcal{L}, S_0)$: "side information" where Σ is the attacker state before challenge, $\mathcal{L}$ is the attacker query/answers to F before challenge , S_0 is the initial PRNG state provided by $\mathcal{A}$;
- U_r: uniform r-bit string.

The advantage of the adversary $\mathcal{A}$ in the recovering game is bounded by $\mathsf{SD}((\tilde{Y}_\ell, Z, F), (U_r, Z, F))$. Therefore, it is sufficient to upper-bound that. The reasoning is quite simple. Note that Z contains the state of the attacker Σ just before it makes the challenge query. This means that $\mathcal{A}$ cannot tell apart real from random.

We also define an event $\mathcal{E}$ where the answers to the F-queries by $\mathcal{A}$ are distinct *when truncated to the first r bits*. Note that there are a maximum of q queries in this list. We would like to point

out that $\mathcal{E}$ has the same probability of occurring in either experiment, since the experiments are identical up to the point when this even is defined. Therefore, by Proposition 3,

$$\mathsf{SD}((\tilde{Y}_\ell, Z, F), (U_r, Z, F)) \;\leq\; \mathsf{SD}((\tilde{Y}_\ell, Z, F)|\mathcal{E}, (U_r, Z, F)|\mathcal{E}) + \frac{q^2}{2^r} \;;$$

For convenience we let $\rho := q^2/2^r$ for the remainder of the proof. In order to bound the statistical distance conditioned on $\mathcal{E}$, we can rewrite the same as as

$$\mathsf{SD}((\tilde{Y}_\ell, Z, F)|\mathcal{E}, (U_r, Z, F)|\mathcal{E}) \;=\; \sum_{z\in\mathcal{E}} \mathsf{P}[Z = z|\mathcal{E}] \cdot \mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;, \tag{3}$$

where $z \in \mathcal{E}$ is to denote that the sum is taken over all side informations $Z = z$, satisfying $\mathcal{E}$.[23] Define $p_z := \mathsf{Pred}(\overline{X}|Z = z)$, and observe that

$$\mathbf{E}_z[p_z] \;=\; \mathsf{Pred}(\overline{X}|Z) \;\leq\; 2^{-\gamma^*} \;,$$

where the latter inequality follows from the assumption $\mathrm{H}_\infty(\overline{X}|Z) \geq \gamma^*$. Moreover,

$$\mathrm{H}_\infty(\overline{X}|Z\mathcal{E}) \;\geq\; \gamma^* - \log(1-\rho)^{-1} \;,$$

which is due to

$$\begin{aligned}
\mathsf{Pred}(\overline{X}|Z) \;&\geq\; \sum_{z\in\mathcal{E}} \mathsf{P}[Z = z] \cdot p_z \\
&=\; \mathsf{P}[\mathcal{E}] \cdot \sum_{z\in\mathcal{E}} \frac{\mathsf{P}[Z = z]}{\mathsf{P}[\mathcal{E}]} \cdot p_z \\
&=\; \mathsf{P}[\mathcal{E}] \cdot \mathsf{Pred}(\overline{X}|Z\mathcal{E}) \;.
\end{aligned}$$

From Lemma 36 we prove below, we will get,

$$\mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;\leq\; \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}}} \;.$$

Using Jensen's inequality, (3) becomes, for $\alpha = 2^r$ and $\beta = \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}}$,

$$\begin{aligned}
\mathsf{SD}((\tilde{Y}_\ell, F)|\mathcal{E}, (U_r, F)|\mathcal{E}) \leq\; & \frac{1}{2}\sqrt{\alpha\mathsf{Pred}(\overline{X}|\mathcal{L}\mathcal{E}) + \beta} \\
& \leq \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 \frac{q^2 2^r}{2^{2n}}} \;.
\end{aligned}$$

□

Lemma 36. *For $z \in \mathcal{E}$ and the random variables as defined earlier,*

$$\mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;\leq\; \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}}} \;.$$

Proof. Fix $z = (\sigma, L, s_0)$. Observe that, conditioned on z, F is distributed uniformly over the set of all functions that agree with L. Thus, by Proposition 2,

$$\mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;\leq\; \frac{1}{2}\sqrt{2^r \cdot \mathsf{Coll}(\tilde{Y}_\ell|Fz) - 1} \;. \tag{4}$$

To bound the collision probability, we consider the following experiment:

[23] Therefore $\mathcal{E}$ can be omitted in the conditioning of the statistical distance.

- choose F uniformly consistent with L
- sample inputs $\overline{X} = (\overline{X}_1, \ldots, \overline{X}_\ell)$ and $\overline{X}' = (\overline{X}'_1, \ldots, \overline{X}'_{\ell'})$ independently but conditioned on $Z = z$.
- compute $\tilde{Y}_\ell$ and $\tilde{Y}'_\ell$ as the truncated MD evaluations with F of $\overline{X}$ and $\overline{X}'$

Then, we bound the probability that $\tilde{Y}_\ell = \tilde{Y}'_\ell$ in this experiment as

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\leq\; \mathsf{P}[\overline{X} = \overline{X}'] + \mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell | \overline{X} \neq \overline{X}'] \; . \tag{5}$$

Clearly, the former is at most p_z. To bound the latter, we fix arbitrary inputs $\overline{x} \neq \overline{x}'$ of lengths ℓ and ℓ', respectively. We also assume, wlog, that the evaluation of $\overline{x}'$ is not completely covered by L; due to the collision-freeness of L. Let $\overline{x}_{k+1}$ be the first block of $\overline{x}$ not covered by $\mathcal{L}$ and similarly $\overline{x}'_{k+1}$ for $\overline{x}_{\prime}$. We let $k = \ell$ if all blocks are covered.

The structure graph. We will now model the evaluation producing $\tilde{Y}_\ell$ and $\tilde{Y}'_\ell$ as a (labeled) graph process. In particular, let us define for convenience

$$x^{(i)} := \begin{cases} \overline{x}_i & i \leq \ell \\ \overline{x}'_{(i-\ell)} & \text{otherwise} \end{cases}$$

for all $i = 1, \ldots, \ell + \ell'$, and we let $\tilde{\ell} = \ell + \ell'$.

For a given compression function F (consistent with L), first define a labeled (multi-)graph $H_F(\overline{x}, \overline{x}_{\prime}) = (\mathcal{V}, \mathcal{E}.\mathcal{L})$ vertex set $\mathcal{V} \subseteq \{0,1\}^n$. Each edge will have a label $\mathcal{L}(e) = (x, c)$ – where $x \in \{0,1\}^m$ and $c \in \{\mathsf{red}, \mathsf{blue}\}$. Now we add edges as follows, allowing replications of edges (we will then explain below how to remove duplicates). We will implicitly define $\mathcal{V}$ as the set of n-bit strings which are endpoint to an edge:

Blue edges. For every $((x, y), y')$ in L we add an edge (y, y') with label (x, blue), and refer to such edges as the *blue* edges.

Red edges. Define

$$s_i := \begin{cases} 0^n & i = 0 \\ F(s_{i-1}, \overline{x}_i) & 1 \leq i \leq \ell \\ F(0^n, \overline{x}'_1) & i = \ell + 1 \\ F\left(s_{i-1}, \overline{x}'_{(i-\ell)}\right) & \ell + 2 \leq i \leq \tilde{\ell} \; . \end{cases}$$

Now for $i = 1, \ldots, \ell$, we add edge (s_{i-1}, s_i) with label $(\overline{x}_i, \mathsf{red})$, unless the edge (s_{i-1}, s_i) is already present with label $(\overline{x}_i, c)$ for $c \in \{\mathsf{blue}, \mathsf{red}\}$. We add the edge $(0^n, s_{\ell+1})$ with label $(\overline{x}'_1, \mathsf{red})$, unless the edge (s_{i-1}, s_i) is already present with label $(\overline{x}'_1, c)$ for $c \in \{\mathsf{blue}, \mathsf{red}\}$. Finally, we add each (s_{i-1}, s_i) for $i = \ell + 2, \ldots, \tilde{\ell}$ with label $(\overline{x}'_i, \mathsf{red})$, unless the edge (s_{i-1}, s_i) is already present with label $(\overline{x}'_i, c)$ for $c \in \{\mathsf{blue}, \mathsf{red}\}$. We refer to these edges we added as the *red* edges.

Note in particular that we may have two identical edges e_1 and e_2, but in this case they will have labels (x_1, c_1) and (x_2, c_2) with $x_1 \neq x_2$, and moreover, at least one of them is red by our assumption on L.

Definition 14. *The* structure graph *of $G_F = G_F(\overline{x},\overline{x}_\prime) = (\mathcal{V},\mathcal{E},\mathcal{L})$ is the graph obtained from H_F as follows. We first look at all isomorphic graphs to G_F with vertices $\{0,\ldots,|\mathcal{V}|-1\}$ such that 0^n is mapped to 0. We then pick the lexicographically first such graph.*[24]

Let $\mathcal{G}_L(\overline{x},\overline{x}')$ be the set of all $G_F(\overline{x},\overline{x}')$ for an F compatible with L. Note that G_F will have two (possibly overlapping) paths starting from 0 with edges labeled by $\overline{x}$ and $\overline{x}'$, containing blue and red edges. We say that G_F is *colliding* if these two paths end in the same vertex, and let $\mathsf{Coll}_L(\overline{x},\overline{x}') \subseteq \mathcal{G}_L(\overline{x},\overline{x}')$ be the set of colliding structure graphs.

Definition 15 (Accidents)**.** *Let now $\mathcal{B}$ be the set of vertices of the sub-graph of G_F induced by the blue edges, plus 0. We now traverse the path induced by $\overline{x}$, and then the path induced by $\overline{x}'$. Each time we encounter a vertex which is not in $\mathcal{B}$, we add it to it. Each time we encounter a vertex which is* already *in $\mathcal{B}$, we say that an* accident *has occurred. We let $\mathsf{Acc}(G_F)$ be the number of accidents in G_F.*

We also let $\mathcal{G}^a_L(\overline{x},\overline{x}')$ to be the set of $H \in \mathcal{G}_L(\overline{x},\overline{x}')$ with $\mathsf{Acc}(H) = a$. We state the following lemmas.

Lemma 37. *Let F be sampled randomly consistent with L, and $\overline{x} \neq \overline{x}'$. Then, let $\tilde{Y}_\ell$ and $\tilde{Y}'_\ell$ be values obtained after truncating at the end of the MD evaluations on inputs $\overline{x}$ and $\overline{x}'$ respectively. Then,*

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \leq \mathsf{P}[G_F(\overline{x},\overline{x}') \in \mathsf{Coll}_L(\overline{x},\overline{x}')] + \frac{1}{2^r}\ .$$

Proof. We rewrite $\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell]$ as:

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \leq \mathsf{P}[G_F(\overline{x},\overline{x}') \in \mathsf{Coll}_L(\overline{x},\overline{x}')] + \mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell | G_F(\overline{x},\overline{x}') \notin \mathsf{Coll}_L(\overline{x},\overline{x}')]$$

We take a closer look at the latter term. Note that the graph is a fixed graph and it has no collision at the end nodes. We proceed to assign random, yet distinct values to the vertices. These are chosen from $\{0,1\}^n$. Note that it is sufficient to look at the two output vertices *locally* without looking at the global state. There are $2^n(2^n-1)$ pairs of values for these output vertices such that these values are distinct. However, of these, $2^r 2^{n-r}(2^{n-r}-1)$ have values which are equal in the first r bits and yet are distinct n-bit strings. Therefore,

$$\begin{aligned}\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell | G_F(\overline{x},\overline{x}') \notin \mathsf{Coll}_L(\overline{x},\overline{x}')] &\leq \frac{2^r 2^{n-r}(2^{n-r}-1)}{2^n(2^n-1)} \\ &= \frac{2^{n-r}-1}{2^n-1} \\ &\leq \frac{2^{n-r}}{2^n} = \frac{1}{2^r}\ .\end{aligned}$$

Putting it together, we have:

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \leq \mathsf{P}[G_F(\overline{x},\overline{x}') \in \mathsf{Coll}_L(\overline{x},\overline{x}')] + \frac{1}{2^r}\ . \tag{6}$$

□

Lemma 38. *Let F be sampled randomly consistent with L, and $\overline{x} \neq \overline{x}'$. Let $H \in \mathcal{G}_L(\overline{x},\overline{x}')$. Then,*

$$\mathsf{P}[G_F(\overline{x},\overline{x}') = H] = \frac{1}{2^{n\cdot\mathsf{Acc}(H)}}\ .$$

[24] Indeed, the actual labeling of graphs will not matter below.

Proof. We have two messages $\overline{x} \neq \overline{x}'$ and let $x^{(i)}$ be as defined before. We have a randomly sampled F which is consistent with L. Let us assume that the values $S_1, \ldots, S_{\tilde{\ell}}$ are revealed to us stepwise. S_i is the random variable representing the vertex after block i. Let $G_F = G_F(\overline{x}, \overline{x}')$. We define a consistency notion as follows: G_F is consistent with a given graph H after step $i \leq \tilde{\ell}$, denoted by Cons_i if the structure graphs $G_F^{(i)}$ and $H^{(i)}$ are equal as triples $(\mathcal{V}, \mathcal{E}.\mathcal{L})$ where $G_F^{(i)}$ is the graph G_F obtained after the first i blocks are processed. We define $H^{(i)}$ similarly. We assume that Cons_i is true for some i. We now bound $\mathsf{P}[\mathsf{Cons}_{i+1}|\mathsf{Cons}_i]$. We look at the step $i+1$ in H and there are three possibilities on how the edge for message block $x^{(i+1)}$ looks:

- *Fresh*: It arrives at a new vertex which is not present in $H^{(i)}$.
- *Determined*: It follows an existing edge, i.e there exists a label for the edge of the form $(x^{(i+1)}, .)$
- *Accident*: It causes an accident. In this case, $G_F^{(i+1)}$ will only be consistent if the edge corresponding to $x^{(i+1)}$ lands on the same vertex as in $H^{(i+1)}$. Note that this accident is a fresh-evaluation, i.e the output is not determined in the first i steps and is therefore chosen randomly from 2^n values. In other words, $\mathsf{P}[\mathsf{Cons}_{i+1}|\mathsf{Cons}_i] = \frac{1}{2^n}$ in this case.

Note that the third case happens $\mathsf{Acc}(H)$ times. Therefore, we have:

$$\mathsf{P}[G_F(\overline{x}, \overline{x}') = H] = \mathsf{P}[\mathsf{Cons}_{\tilde{\ell}}] \leq \frac{1}{2^{n \cdot \mathsf{Acc}(H)}} .$$

The latter inequality arises by upper-bounding $\mathsf{P}[\mathsf{Cons}_{i+1}|\mathsf{Cons}_i]$ with 1 in the case of Fresh and Determined edges. □

Lemma 39. *Let F be sampled randomly consistent with L, and $\overline{x} \neq \overline{x}'$. Then,*

$$\mathsf{P}[\mathsf{Acc}(G_F) \geq 2] \leq \frac{64\ell^4}{2^{2n}} + \frac{16\ell^2 q^2}{2^{2n}} .$$

Proof. We define $\mathcal{G}_L^a$, as before, to be the set of all structure graphs containing exactly a accidents.

$$\begin{aligned}
\mathsf{P}[\mathsf{Acc}(G_F) \geq 2] &= \sum_{a=2}^{\infty} \mathsf{P}[\mathsf{Acc}(G_F) \geq 2] \\
&= \sum_{a=2}^{\infty} \sum_{g \in \mathcal{G}_L^a} \mathsf{P}[G_F = g] \\
&\leq \sum_{a=2}^{\infty} \frac{|\mathcal{G}_L^a|}{2^{n \cdot a}} .
\end{aligned}$$

The last step follows from Lemma 38. Observe that for fixed base graph and fixed messages, entire graph is defined by list of accidents. Each accident is defined by an edge (i, j), where there are $\tilde{\ell} := \ell + \ell'$ choices for i and $\tilde{\ell} + q$ for j. Thus, there are at most $(\tilde{\ell}(\tilde{\ell} + q))^c$ graphs with a accidents.

$$\begin{aligned}
\mathsf{P}[\mathsf{Acc}(G_F) \geq 2] &\leq \sum_{a=2}^{\infty} \frac{|\mathcal{G}_L^a|}{2^{n \cdot a}} \\
&\leq \sum_{a=2}^{\infty} \left(\frac{\tilde{\ell}(\tilde{\ell} + q)}{2^n} \right)^a \\
&\leq \left(\frac{\tilde{\ell}(\tilde{\ell} + q)}{2^n} \right)^2 \leq \frac{4\tilde{\ell}^4}{2^{2n}} + \frac{4\tilde{\ell}^2 q^2}{2^{2n}} \leq \frac{64\ell^4}{2^{2n}} + \frac{16\ell^2 q^2}{2^{2n}} .
\end{aligned}$$

where we assume $\tilde{\ell} \leq 2\ell$ □

Note that $H \in \mathsf{Coll}_L = \mathsf{Coll}_L(\overline{x}, \overline{x}')$ necessarily implies $\mathsf{Acc}(H) \geq 1$. We can then say, for $G_F = G_F(\overline{x}, \overline{x}')$,

$$\begin{aligned} \mathsf{P}[G_F \in \mathsf{Coll}_L] &\leq \mathsf{P}[G_F \in \mathsf{Coll}_L \wedge \mathsf{Acc}(G_F) = 1] + \mathsf{P}[\mathsf{Acc}(G_F) \geq 2] \\ &\leq \frac{\left|\mathsf{Coll}_L \cap \mathcal{G}_L^1\right|}{2^n} + \frac{64\ell^4}{2^{2n}} + \frac{16\ell^2 q^2}{2^{2n}} \end{aligned} . \tag{7}$$

by Lemmas 38 and 39. We are going to upper bound the number of graphs in $\mathsf{Coll}_L \cap \mathcal{G}_L^1$.

We reduce handling the case of a general L now to the simpler case where $L = \emptyset$, so that we can resort to the counting argument of [24].

Now, let $G \in \mathsf{Coll}_L \cap \mathcal{G}_L^1(\overline{x}, \overline{x}')$. Let $\mathcal{E}_B$ be the set of all blue edges which are on the $\overline{x}$- and the $\overline{x}'$-paths. Moreover, let $\mathcal{E}'_B \subseteq \mathcal{E}_B$ be the set of all *initial* blue edges, i.e., the set of blue edges on these paths which are not preceded by any red edge. We prove the following lemma.

Lemma 40. *If $G \in \mathsf{Coll}_L \cap \mathcal{G}_L^1(\overline{x}, \overline{x}')$, then $\mathcal{E}'_B = \mathcal{E}_B$*

Proof. We prove by contradiction by assuming that this is not true. In other words, there exists at least one edge $e \in \mathcal{E}_B \setminus \mathcal{E}'_B$. We now proceed to show that this necessarily leads to at least two collisions, contradicting $G \in \mathsf{Coll}_L \cap \mathcal{G}_L^1(\overline{x}, \overline{x}')$. Wlog, assume that this lies on the $\overline{x}$ path, and let $e_i = (u_i, v_i)$ be the first such edge. Then, this edge must be preceded by a red-edge, e_{i-1}. Then, clearly this implies that the graph contains at least one accident.

Now, let us take a look at the $\overline{x}'$ path. We have, by our definition of G (it is in the set Coll_L), that there should exist a path from e_i to the end point of the $\overline{x}'$ path. We will sketch a proof to show that this is not possible without having a second collision. We take a look at this path from e_i to the end point of $\overline{x}'$, V'.

- The path is of entirely blue edges.

  We look at the edge into V' on the $\overline{x}'$ path. Note that this cannot be a new blue edge as it would violate the no-collision constraint on L. If this was a red edge we have a second collision, by our definition. The only option is to take the same blue edge as on the e_i-path. In other words, the penultimate vertex on e_i-path and $\overline{x}'$ path are the same. We can argue similarly for the penultimate vertex by showing that the edges into them should be the same blue edge on both the paths. Working backwards we arrive at the point u_i. Thus, we have shown that the message blocks corresponding to the edges along the path from e_i to V' is the same on both $\overline{x}$ and $\overline{x}'$. We have also assumed that e_i is the first non-initial blue edge. In other words, the path before e_i should be all red. Since $\overline{x} \neq \overline{x}'$, there should be a block before e_i which would differ in the message and hence this would constitute a collision on the red edges. Therefore, in this case it is impossible to have the final outputs to be the same without causing a second collision.

- The path is of entirely red edges.

  The argument is similar to the previous case. The only difference is that the $\overline{x}'$-path could take a blue edge into the e_i-path. However, this would still constitute a new collision. The same reasoning follows giving us the same conclusion.

- The path is mixed red and blue edges.

  We have already shown that if a blue edge follows a red edge on the path then it is a collision. We can therefore assume that the set of blue edges occur together, followed by the set of red edges. The reasoning is pretty similar to the previous two cases.

In other words, a graph G cannot be in Coll_L and $\mathcal{G}^1_L(\overline{x}, \overline{x}')$ when there exist non-initial blue edges. □

The direct consequence of Lemma 40 is that the number of colliding structure graphs with one accident is the same even if we remove all the non-initial blue edges. In addition, removing the initial blue edges will also not decrease the number of colliding graphs with one accident. This is true because the initial blue edges can be replaced by red edges without increasing the number of accidents by the properties of L. In other words, if we define $L' \subseteq L$ to be the set of queries which define the initial blue edges, we have

$$\left|\mathsf{Coll}_L \cap \mathcal{G}^1_L(\overline{x}, \overline{x}')\right| = \left|\mathsf{Coll}_{L'} \cap \mathcal{G}^1_{L'}(\overline{x}, \overline{x}')\right| \leq \left|\mathsf{Coll}_{\emptyset} \cap \mathcal{G}^1_{\emptyset}(\overline{x}, \overline{x}')\right| \leq \ell d'(\ell) \ . \tag{8}$$

where the last inequality is from [24]. Here $d'(n)$ is defined as follows:

$$d'(n) := \max_{n' \in \{1,\ldots,n\}} \left|\left\{d \in \mathbb{N} : n' \mod d = 0\right\}\right| \ .$$

Combining Equations 6, 7 and 8 we get,

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \leq \frac{1}{2^r} + \frac{\ell \cdot d'(\ell)}{2^n} + \frac{64\ell^4}{2^{2n}} + \frac{16\ell^2 q^2}{2^{2n}}$$

Substituting the above value in Equations 13 and 14, we get:

$$\mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \ \leq \ \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}}} \ .$$

This concludes the proof of Lemma 36 □

7.5 IT PRNGs from Merkle-Damgård with Davies-Meyer

This section establishes the robustness of the PRNG construction $\mathsf{DM}_r = (\mathsf{refresh}, \mathsf{next})$ based on Merkle-Damgård with the Davies-Meyer compression function in the ideal-cipher model. Recall that the construction is defined as follows (cf. Figure 4):

Construction 5 (IT-PRNG from MD-DM)**.** The (n, r)-PRNG construction $\mathsf{DM}_r = (\mathsf{refresh}, \mathsf{next})$ using Merkle-Damgård with Davies-Meyer (MD-DM) uses a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and is defined as follows:

- $\mathsf{refresh}^E(s, x) = E(x, s) \oplus s$, and
- $\mathsf{next}^E(s) = (0^n, s[1..r])$.

Construction 5 achieves the following security:

Theorem 7 (IT-Robustness of MD-DM PRNGs)**.** *Construction 5 is a $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$-IT-robust PRNG in the E-model, where*

$$\varepsilon_{\text{rob-it}} \ \leq \ \frac{t}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + 64\ell^4 \cdot \frac{2^r}{2^{2n-2}} + 16\ell^2\tilde{q}^2 \cdot \frac{2^r}{2^{2n-2}}} + t\rho \ ,$$

for $\rho = \frac{\tilde{q}^2}{2^r}$ where $\tilde{q} = q + t\ell$

Our robustness proof for this construction uses the analysis from Section 7.4. The intuition here is that the structure graphs were defined for a compressing function F and we instantiate it with a Davies-Meyer construction and extend the arguments. However, there is a subtle difference because of the underlying primitive. In Davies-Meyer the primitive is an ideal cipher. Thus, we state and prove Lemma 43 which is the counterpart of Lemma 38 for the Davies-Meyer instantiation. We then apply the results of Lemma 43 in the proofs of Lemma 39 and results from [24] to prove Theorem 6.

Proof. The proof follows from Lemma 41 and Theorem 34. □

Lemma 41. *For every γ^*-IT-legitimate (q,ℓ)-attacker, in the ideal cipher model,*

$$\mathrm{Adv}^{\text{rec-IT},\gamma^*}_{\mathsf{DM}}(\mathcal{A}) \;\leq \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + \frac{64\ell^4 2^r}{2^{2n-2}} + \frac{16\ell^2 q^2 2^r}{2^{2n-2}}} + \rho \;.$$

where $\rho = \frac{q^2}{2^r}$

Proof. The arguments for this proof is similar to the proof of Lemma 35. As before we define the random variables $\tilde{Y}_\ell, Z, \Sigma, U_r$. The difference arises in the definition of $Z = (\Sigma, \mathcal{L}, S_0)$. We define the $\mathcal{L}$ as follows: If $\mathcal{L}'$ is the set of queries made by $\mathcal{A}$ to the ideal cipher E, then for every $((x,y),z) \in \mathcal{L}'$, add $((x,y), z \oplus y)$ to $\mathcal{L}$. This is to ensure that the $\mathcal{L}$ is consistent with the evaluation of Davies-Meyer on the input (x,y). We now upper-bound the adversary $\mathcal{A}$ in the IT-recovering game by showing an upperbound for $\mathsf{SD}((\tilde{Y}_\ell, Z, E), (U_r, Z, E))$.

We also define the event $\mathcal{E}$ which denotes the event when $\mathcal{L}$ entries are distinct *when truncated to the first r bits*. Note that there are a maximum of q queries in this list. We would like to point out that $\mathcal{E}$ has the same probability of occurring in either experiment. Therefore, by Proposition 3,

$$\mathsf{SD}((\tilde{Y}_\ell, Z, E), (U_r, Z, E)) \;\leq\; \mathsf{SD}((\tilde{Y}_\ell, Z, E)|\mathcal{E}, (U_r, Z, E)|\mathcal{E}) + \frac{q^2}{2^r} \;;$$

We let $\rho := q^2/2^r$ for the remainder of the proof. In order to bound the statistical distance conditioned on $\mathcal{E}$, we can rewrite the same as

$$\mathsf{SD}((\tilde{Y}_\ell, Z, E)|\mathcal{E}, (U_r, Z, E)|\mathcal{E}) \;=\; \sum_{z \in \mathcal{E}} \mathsf{P}[Z = z|\mathcal{E}] \cdot \mathsf{SD}((\tilde{Y}_\ell, E)|z, (U_r, E)|z) \;, \tag{9}$$

where $z \in \mathcal{E}$ is to denote that the sum is taken over all side informations $Z = z$, satisfying $\mathcal{E}$.[25] We define $p_z := \mathsf{Pred}(\overline{X}|Z = z)$, and as shown in Lemma 35, we have

$$\mathrm{H}_\infty(\overline{X}|Z\mathcal{E}) \;\geq\; \gamma^* - \log(1-\rho)^{-1} \;,$$

From Lemma 42 we get,

$$\mathsf{SD}((\tilde{Y}_\ell, E)|z, (U_r, E)|z) \leq \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + 64\ell^4\frac{2^r}{2^{2n-2}} + 16\ell^2 q^2 \frac{2^r}{2^{2n-2}}}$$

Using Jensen's inequality, (9) becomes, for $\alpha = 2^r$ and $\beta = \ell \cdot d'(\ell) \cdot \frac{2^r}{2^{n-1}} + 64\ell^4 \cdot \frac{2^r}{2^{2n-2}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n-2}}$,

$$\begin{aligned}\mathsf{SD}((\tilde{Y}_\ell, E)|\mathcal{E}, (U_r, E)|\mathcal{E}) \leq\; & \frac{1}{2}\sqrt{\alpha \mathsf{Pred}(\overline{X}|\mathcal{L}\mathcal{E}) + \beta} \\ & \leq \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + 64\ell^4\frac{2^r}{2^{2n-2}} + 16\ell^2\frac{q^2 2^r}{2^{2n-2}}} \;.\end{aligned}$$

□

[25] Therefore $\mathcal{E}$ can be omitted in the conditioning of the statistical distance.

Lemma 42. *For $z \in \mathcal{E}$ and the random variables as defined earlier,*

$$\mathsf{SD}((\tilde{Y}_\ell, E)|z, (U_r, E)|z) \leq \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + 64\ell^4\frac{2^r}{2^{2n-2}} + 16\ell^2 q^2 \frac{2^r}{2^{2n-2}}}$$

Proof. We fix $z = (\sigma, L, s_0)$. We have that z, E is distributed uniformly over the set of all ideal ciphers that agree with L. Thus, by Proposition 2,

$$\mathsf{SD}((\tilde{Y}_\ell, E)|z, (U_r, E)|z) \;\leq\; \frac{1}{2}\sqrt{2^r \cdot \mathsf{Coll}(\tilde{Y}_\ell|Ez) - 1}\;. \tag{10}$$

To bound the collision probability, we consider the following experiment:

- choose E uniformly consistent with L
- sample inputs $\overline{X} = (\overline{X}_1, \ldots, \overline{X}_\ell)$ and $\overline{X}' = (\overline{X}'_1, \ldots, \overline{X}'_{\ell'})$ independently but conditioned on $Z = z$.
- compute $\tilde{Y}_\ell$ and $\tilde{Y}'_\ell$ as the truncated MD evaluations with E of $\overline{X}$ and $\overline{X}'$

Now, we have that:

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\leq\; \mathsf{P}[\overline{X} = \overline{X}'] + \mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell | \overline{X} \neq \overline{X}']\;. \tag{11}$$

Clearly, the former is at most p_z. To get a bound on the second term, we proceed to fix arbitrary inputs $\overline{x} \neq \overline{x}'$ of lengths ℓ and ℓ', respectively. This is similar to the process that is adopted in the proof of Lemma 36. We construct a similar Structure Graph. Note that in Lemma 36, we assumed that the primitive was a compressing function. In particular, Davies Meyer is also a compressing function. Therefore, we can define similar structure graph for this construction by viewing $F(y, x) := E(x, y) \oplus y$. From Lemma 37 we get the following result:

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \leq \mathsf{P}[G_F(\overline{x}, \overline{x}') \in \mathsf{Coll}_L(\overline{x}, \overline{x}')] + \frac{1}{2^r}\;.$$

However, the proof of Lemma 38 changes. We prove a corresponding Lemma as follows:

Lemma 43. *For an E sampled randomly consistent with L with F defined as above, and $\overline{x} \neq \overline{x}'$. Let $H \in \mathcal{G}_L(\overline{x}, \overline{x}')$. Then,*

$$\mathsf{P}[G_F(\overline{x}, \overline{x}') = H] = \frac{1}{2^{(n-1)\cdot \mathsf{Acc}(H)}}\;.$$

Proof. We have two messages $\overline{x} \neq \overline{x}'$ and let $x^{(i)}$ be as defined before. We have a randomly sampled E which is consistent with L. The proof is similar to Lemma 38. However, the proof differs in the case of a colliding edge, i.e the edge causes an accident. In this case, $G_F^{(i+1)}$ will only be consistent if the edge corresponding to $x^{(i+1)}$ lands on the same vertex as in $H^{(i+1)}$. However, the evaluation is not a random function F which chooses a value, uniformly at random, from 2^n values. The evaluation is dependent on the definition of the ideal cipher E. The output in this case is chosen, from a minimum of $2^n - q - i$ values. In other words, $\mathsf{P}[\mathsf{Cons}_{i+1}|\mathsf{Cons}_i] \leq \frac{1}{2^n - q - i} \leq \frac{1}{2^{(n-1)}}$ in this case. Note that the third case happens $\mathsf{Acc}(H)$ times. Therefore, we have:

$$\mathsf{P}[G_F(\overline{x}, \overline{x}') = H] = \mathsf{P}[\mathsf{Cons}_{\tilde{\ell}}] \leq \frac{1}{2^{(n-1)\cdot \mathsf{Acc}(H)}}\;.$$

□

Applying this Lemma to the proof of Lemma 39 and the result from [24], we get:

$$\mathsf{SD}((\tilde{Y}_\ell, E)|z, (U_r, E)|z) \leq \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + 64\ell^4\frac{2^r}{2^{2n-2}} + 16\ell^2 q^2 \frac{2^r}{2^{2n-2}}}\;.$$

□

References

[1] 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, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press.

[2] Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. In *45th FOCS*, pages 384–393, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[3] Boaz Barak, Ronen Shaltiel, and Eran Tromer. True random number generators secure in a changing environment. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 166–180, Cologne, Germany, September 8–10, 2003. Springer, Heidelberg, Germany.

[4] Elaine Barker and John Kelsey. NIST Special Publication 800-90A (A revision of SP 800-90) Recommendation for random number generation using deterministic random bit generators. https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final, 2012.

[5] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. NIST Special Publication 800-90A, 2012.

[6] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[7] Mihir Bellare, Krzysztof Pietrzak, and Phillip Rogaway. Improved security analyses for CBC MACs. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 527–545, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[8] 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, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[9] 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, Santa Barbara, CA, USA, August 17–20, 2010. Springer, Heidelberg, Germany.

[10] Manuel Blum. Independent unbiased coin flips from a correlated biased source-a finite stae markov chain. *Combinatorica*, 6(2):97–108, 1986.

[11] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 670–683, Cambridge, MA, USA, June 18–21, 2016. ACM Press.

[12] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 319–335, 2014.

[13] 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, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[14] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity (extended abstract). In *26th FOCS*, pages 429–442, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press.

[15] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

[16] Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *26th FOCS*, pages 396–407, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press.

[17] 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, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[18] 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, Berlin, Germany, November 4–8, 2013. ACM Press.

[19] 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, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

[20] 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, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[21] D. Eastlake, J. Schiller, and S. Crocker. *RFC 4086 - Randomness Requirements for Security*, June 2005.

[22] Niels Ferguson. Private communication, 2013.

[23] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2003.

[24] Peter Gaži, Krzysztof Pietrzak, and Michal Rybár. The exact PRF-security of NMAC and HMAC. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 113–130, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[25] Peter Gazi and Stefano Tessaro. Provably robust sponge-based PRNGs and KDFs. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 87–116, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[26] Danny Harnik and Moni Naor. On everlasting security in the hybrid bounded storage model. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 192–203, Venice, Italy, July 10–14, 2006. Springer, Heidelberg, Germany.

[27] Daniel Hutchinson. A robust and sponge-like PRNG with improved efficiency. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 381–398, St. John's, NL, Canada, August 10–12, 2016. Springer, Heidelberg, Germany.

[28] Daniel Hutchinson. A robust and sponge-like PRNG with improved efficiency. Cryptology ePrint Archive, Report 2016/886, 2016. http://eprint.iacr.org/2016/886.

[29] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[30] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th FOCS*, pages 230–235, Research Triangle Park, North Carolina, October 30 – November 1, 1989. IEEE Computer Society Press.

[31] Information technology - Security techniques - Random bit generation. ISO/IEC18031:2011, 2011.

[32] Jesse Kamp, Anup Rao, Salil P. Vadhan, and David Zuckerman. Deterministic extractors for small-space sources. *J. Comput. Syst. Sci.*, 77(1):191–220, 2011.

[33] John Kelsey, Bruce Schneier, and Niels Ferguson. Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In *In Sixth Annual Workshop on Selected Areas in Cryptography*, pages 13–33. Springer, 1999.

[34] Killmann, W. and Schindler, W. A proposal for: Functionality classes for random number generators. AIS 20 / AIS31, 2011.

[35] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[36] David Lichtenstein, Nathan Linial, and Michael E. Saks. Some extremal problems arising form discrete control processes. *Combinatorica*, 9(3):269–287, 1989.

[37] John M. Intel digital random number generator (DRNG) software implementation guide. https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide, 2014.

[38] Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 235–244, 1993.

[39] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[40] 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, Sackville, New Brunswick, Canada, August 14–15, 2009. Springer, Heidelberg, Germany.

[41] Berry Schoenmakers and Andrey Sidorenko. Cryptanalysis of the dual elliptic curve pseudorandom generator. Cryptology ePrint Archive, Report 2006/190, 2006. http://eprint.iacr.org/2006/190.

[42] 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, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[43] Dan Shumow and Niels Ferguson. On the possibility of a back door in the nist sp800-90 dual ec prng. *CRYPTO Rump Session*, 2007.

[44] Pratik Soni and Stefano Tessaro. Public-seed pseudorandom permutations. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 412–441, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[45] Luca Trevisan and Salil P. Vadhan. Extracting randomness from samplable distributions. In *41st FOCS*, pages 32–42, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[46] John von Neumann. Various techniques used in connection with random digits. In A.S. Householder, G.E. Forsythe, and H.H. Germond, editors, *Monte Carlo Method*, pages 36–38. National Bureau of Standards Applied Mathematics Series, 12, Washington, D.C.: U.S. Government Printing Office, 1951.

[47] Wikipedia. /dev/random. http://en.wikipedia.org/wiki//dev/random, 2004. [Online; accessed 09-February-2014].

[48] Joanne Woodage and Dan Shumow. An analysis of the NIST SP 800-90A standard. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19 - May 23*, 2019.

A Comparison to Previous PRNG Security Notions

Overview. The first robustness notion for PRNGs was proposed by Barak and Halevi [1] and required that PRNGs be:

- *resilient*, i.e., the attacker cannot predict the output of the PRNG even if he can influence the inputs;
- *forward secure*, i.e., upon state compromise, previous outputs of the PRNG remain random; and
- *post-compromise secure*, i.e., if sufficient entropy is absorbed into the PRNG state, the outputs regain security.

The definition was developed further in a crucial way by Dodis *et al.* [18], who considered the setting where entropy only *gradually* accumulates in the state (as opposed to the model in [1], where inputs are required to have high entropy in order to recover from compromise. In order to analyse constructions such as sponges, Gazi and Tessaro [25] extended the robustness definition further to a setting with a public random permutation.

Distribution samplers. In order to capture distributions for which the PRNG is expected to be robust, Dodis *et al.* [18] considered distribution samplers Sam, which with every input x_i for the PRNG would also provide an entropy estimate γ_i and a leakage value z_i. A sampler was considered *legitimate* if every input x_i has the promised min entropy γ_i even conditioned on all *other* inputs (i.e., past and future) as well as on *all* values γ_j and z_j. A very important consequence of this definition is that (it can be shown that) extraction with such a sampler Sam is impossible without a randomly chosen seed about which the sampler has no information. Dodis *et al.* provided a construction that uses such a seed and whose refresh function actually is information-theoretically secure. However, security completely breaks down as soon as the sampler can depend on the seed, which is why said construction was deemed unsuitable by practitioners since seed-independence of the input may not be guaranteed in practice.

In an effort to analyze more practical constructions Gazi and Tessaro [25] extended the robustness notion to incorporate ideal primitives. In particular, for their analysis of Sponge, they considered samplers Sam^{π} with oracle access to a random permutation π and required an entropy notion related to that above, but required, informally, that inputs be unpredictable even if in addition to the above, one also conditions on the queries by the sampler (albeit the sampler is allowed *private* queries for every input it generates). While the extension to ideal models allows to analyze constructions used in practice, the work by Gazi and Tessaro does not solve the need for a seed (about which Sam^{π} has no information), without which the aforementioned impossibility still holds.

Main differences between existing notions and the new one. The new definition put forth in this work differs from previous ones in several ways: First, the legitimacy notion is such that there is no need for a seed. Second, because of that, the sampler and the distinguisher are merged into a single attacker, which would not be possible with the previous notions due to the need to hide the seed from Sam. Third, the new legitimacy notion measures not the sum of entropies of particular blocks conditioned on each other, but the entropy of an entire sequence of inputs *as a whole*. Finally, the new notion dispenses with the need for entropy estimates by simply only considering attackers that provide inputs for which the average min-entropy at certain points is high enough conditioned on the state and the queries of the attacker. Observe that the entropy estimates

were originally introduced to circumvent the impossibility of randomness estimation, which would have been needed in reductions to computationally secure primitives (such as using a PRG for the next function). Hence, entropy estimates were an unnatural artifact of the definition anyway.

B Constructions of Online Extractors

This section presents three simple, intuitive, and—most importantly—practical online-extractor constructions:

- a construction based on the *Merkle-Damgård paradigm* using a public *fixed-length compression function*;
- a construction based on the *Merkle-Damgård paradigm* using the *Davies-Meyer compression function* (as in SHA-2), which is built from any public block cipher; and
- a construction based on the *Sponge paradigm* (as in SHA-3), which uses a public permutation.

For extractors based on the MD paradigm, there are in fact two constructions: one achieving normal, computational security and one achieving information-theoretic (IT) security. As with PRNGs, there is also an IT candidate based on Sponges, but its security analysis is left for future work.

B.1 Extractors from Merkle-Damgård

An online extractor can be obtained from a compression function F as follows:[26]

Construction 8 (Online extractor from Merkle-Damgård)**.** The (m, n)-online extractor construction $\mathsf{MD} = (\mathsf{refresh}, \mathsf{finalize})$ based on Merkle-Damgård with a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is defined as follows:

- $\mathsf{refresh}^F(s, x) = F(s, x)$, and
- $\mathsf{finalize}^F(s) = s$.

The security of Construction 8 is proved in the F-model, where F is a uniformly random function. The theorem below is identical to Lemma 13.

Theorem 44 (Online extractor from Merkle-Damgård)**.** *Construction 8 is a $(\gamma^*, q, \varepsilon)$-online extractor in the F-model for*

$$\varepsilon \;\leq\; \frac{q^2 + q\ell + \ell^2}{2^n} + \frac{2q}{2^{\gamma^*}} \;.$$

An IT-secure online extractor based on Merkle-Damgård can be obtained if the finalize function simply truncates the state:

Construction 9 (IT online extractor from Merkle-Damgård)**.** The (m, r)-IT-online extractor construction $\mathsf{MD}_r = (\mathsf{refresh}, \mathsf{finalize})$ based on Merkle-Damgård with a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is defined as follows:

- $\mathsf{refresh}^F(s, x) = F(s, x)$, and

[26] To reduce notational clutter, the algorithms refresh and finalize of the extractor constructions are not "branded" with the design name. There will be no ambiguity as to which construction is meant in any place in this paper.

- $\mathsf{finalize}^F(s) = s[1..r]$.

The security of Construction 9 is proved in the F-model, where F is a uniformly random function. To state the theorem for the IT construction, for an integer ℓ, let

$$d'(\ell) = \max_{\ell' \in \{1,\ldots,\ell\}} \left|\{d \in \mathbb{N} \ : \ d|\ell'\}\right| .$$

Observe that, asymptotically, $d'(\ell)$ grows very slowly, i.e., as $\ell^{o(1)}$. Furthermore, let F be a random compression function. The following theorem is equivalent to Lemma 35.

Theorem 45 (IT online extractor from Merkle-Damgård)**.** *Construction 9 is a $(\gamma^*, q, \varepsilon)$-IT-online extractor in the F-model, where*

$$\varepsilon \ \leq \ \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 \frac{q^2 2^r}{2^{2n}}} + \rho \quad ,$$

where $\rho = \frac{q^2}{2^r}$.

B.2 Extractors from Merkle-Damgård with Davies-Meyer

The Davies-Meyer compression function maps two inputs $a \in \{0,1\}^m$ and $b \in \{0,1\}^n$ to an n-bit string

$$E(b, a) \oplus a \ ,$$

where E is an arbitrary block cipher (where b is the key and a the input).[27] Correspondingly, an online extractor can be obtained from E as follows:

Construction 10 (Online extractor from MD-DM)**.** The (k, n)-online extractor construction $\mathsf{DM} = (\mathsf{refresh}, \mathsf{finalize})$ based on Merkle-Damgård with Davies-Meyer (MD-DM) uses a cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and is defined as follows:

- $\mathsf{refresh}^E(s, x) = E(x, s) \oplus s$, and
- $\mathsf{finalize}^F(s) = s$.

The security of Construction 10 is proved in the E-model, where E is a cipher chosen uniformly at random from the set of all ciphers and can be queried in both the forward and backward direction. The theorem below is identical to Lemma 20

Theorem 46 (Online extractor from MD-DM)**.** *Construction 10 is a $(\gamma^*, q, \varepsilon_{\text{rob}})$-robust online extractor in the E-model for*

$$\varepsilon_{\text{rob}} \ \leq \ \frac{q^2 + 2(q\ell + \ell^2)}{2^n} + \frac{4q}{2^{\gamma^*}} \ .$$

An IT-secure online extractor based on MD-DM can be obtained if the finalize function simply truncates the state:

Construction 11 (IT online extractor from MD-DM)**.** The (k, r)-IT-online construction $\mathsf{DM}_r = (\mathsf{refresh}, \mathsf{finalize})$ using Merkle-Damgård with Davies-Meyer (MD-DM) uses a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and is defined as follows:

[27] A (block) cipher is an efficiently computable and invertible permutation $E(k, \cdot) : \{0,1\}^n \to \{0,1\}^n$ for every key $k \in \{0,1\}^n$.

- $\mathsf{refresh}^E(s,x) = E(x,s) \oplus s$, and
- $\mathsf{finalize}^E(s) = s[1..r]$.

The security of Construction 11 is proved in the E-model, where E is a cipher chosen uniformly at random from the set of all ciphers and can be queried in both the forward and backward direction. Let $d'(\ell)$ be defined as in Section 5.1. The following theorem is equivalent to Lemma 41.

Theorem 47 (IT online extractor from MD-DM)**.** *Construction 11 is a $(\gamma^*, q, \varepsilon_{\text{rob}})$-IT-online extractor in the E-model, where*

$$\mathrm{Adv}_{\mathsf{DM}}^{\text{rec-IT},\gamma^*}(\mathcal{A}) \;\leq \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell)\frac{2^r}{2^{n-1}} + \frac{64\ell^4 2^r}{2^{2n-2}} + \frac{16\ell^2 q^2 2^r}{2^{2n-2}}} + \rho \;.$$

where $\rho = \frac{q^2}{2^r}$.

B.3 Extractors from Sponges

Let $n \in \mathbb{N}$ and $n = r + c$. In the following, for an n-bit string s, let $s = s^{(\mathsf{r})} \| s^{(\mathsf{c})}$ be decomposition of s into an r-bit and c-bit string. An online extractor using the Sponge paradigm can be obtained from a permutation π as follows:

Construction 12 (Online extractor from Sponges)**.** The Sponge-based online-extractor construction $\mathsf{Spg} = (\mathsf{refresh}, \mathsf{finalize})$ uses a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ to absorb and produce r-bit inputs and outputs, respectively, and is defined as follows:

- $\mathsf{refresh}^\pi(s,x) = \pi(s \oplus x \| 0^c)$, and
- $\mathsf{finalize}^\pi(s) = s[1..r]$.

Observe that for Sponge-based extractors, even the computational variant needs to truncate the state, otherwise the output of the extractor could be inverted by the attacker, which renders the constructions insecure.

The security of Construction 12 is proved in the π-model, where π is a uniformly random permutation, which can be queried in both the forward and backward direction. The proof of the following theorem follows along similar lines as that of Lemma 32.

Theorem 48 (Online extractor from Sponges)**.** *Construction 12 is a $(\gamma^*, q, \ell, \varepsilon)$-online extractor in the π-model for*

$$\varepsilon \;\leq\; 2 \cdot \left(\frac{q + q\ell + \ell^2}{2^n} + \frac{q^2}{2^c} + \frac{q}{2^{\gamma^*}}\right) \;.$$

Observe that the bound in Theorem 8 is only reasonable when c is large enough, which matches the fact that CBC-based online extractors—which correspond to the case $c = 0$, are not secure.

The IT variant of the Sponge construction is identical to the computational variant. Proving its IT-security is left as an interesting open problem.

B.4 Extractors from HMAC

In practice, uniformly random key material is often derived from high-entropy inputs (resulting, e.g., from a key-agreement protocol) using a *key-derivation function (KDF)*. A common paradigm, suggested by Krawczyk [35], to construct KDFs is to combine an extractor with a variable-length pseudorandom function (VL-PRF). The most widely used KDF is *HKDF*, which uses the HMAC mode of operation for compression functions (CF) to instantiate both the extractor and the VL-PRF. This section considers the security of HMAC as a *seedless extractor* w.r.t. the new legitimacy condition put forth by this work. Together with a VL-PRF, the seedless HMAC extractor can then be used to build a KDF. A full treatment of seedless KDFs is deferred to future work.

HMAC. HMAC is similar to Merkle-Damgård, but requires additional CF calls, designed to prevent extension attacks when HMAC is used as a PRF. Concretely, for a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, the HMAC construction takes a key $k \in \{0,1\}^m$ as well as inputs $x_1, \ldots, x_\ell \in \{0,1\}^m$ and outputs[28]

$$\mathsf{HMAC}(k, x_1, \ldots, x_\ell) \;:=\; \mathsf{MD}^F(k \oplus \mathsf{opad}, \mathsf{MD}^F(k \oplus \mathsf{ipad}, x_1, \ldots, x_\ell)) \;,$$

where $\mathsf{ipad} \neq \mathsf{opad} \in \{0,1\}^m$ are arbitrary constants. HMAC can be used as seedless extractor by fixing, say, $k := 0$. That is, the online extractor based on HMAC would be defined as follows:

Construction 13 (Online extractor from HMAC.)**.** The (m, n)-online extractor construction $\mathsf{HMAC} = (\mathsf{refresh}, \mathsf{finalize})$ based on HMAC with a compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is defined as follows:

- the initial state is $s_0 = F(0, \mathsf{ipad})$;
- $\mathsf{refresh}^F(s, x) = F(s, x)$; and
- $\mathsf{finalize}^F(s) = F(F(0, \mathsf{opad}), s)$.

The security of Construction 8 is proved in the F-model, where F is a uniformly random function.

Theorem 49 (Online extractor from HMAC)**.** *Construction 13 is a $(\gamma^*, q, \varepsilon)$-online extractor in the F-model for*

$$\varepsilon \;\leq\; \frac{q^2 + q + q\ell + \ell^2}{2^n} + \frac{2q}{2^{\gamma^*}} \;.$$

Proof (sketch). To prove the security of HMAC as an extractor according to Definition 3, one first considers a hybrid experiment, in which the output of the extractor is computed as

$$F(F(0, \mathsf{opad}), U) \;,$$

for a value chosen uniformly at random. The indistinguishability of the original extraction game and the hybrid experiment is established via a simple hybrid argument: essentially, the reduction (to the security of MD as an extractor) simply prepends a block ipad to the inputs $x_1, \ldots, x_\ell$, and upon receiving a value y, it additionally computes $F(F(0, \mathsf{opad}), y)$. It is easily seen that adding ipad to the input does not affect the conditional entropy of the input—and therefore the legitimacy of the reduction.

Finally, it is easily seen that in the hybrid experiment, the advantage of any attacker is zero unless it queries $F(F(0, \mathsf{opad}), U)$, which happens with probability at most $q/2^n$. □

[28] The IV to the Merkle-Damgård construction is 0.

An IT-secure online extractor based on Merkle-Damgård can be obtained if the finalize function additionally truncates the output:

Construction 14 (IT online extractor from HMAC.)**.** The (m, r)-IT-online extractor construction $\mathsf{HMAC}' = (\mathsf{refresh}, \mathsf{finalize})$ based on HMAC with a compression function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is defined as follows:

- the initial state is $s_0 = F(0, \mathsf{ipad})$;
- $\mathsf{refresh}^F(s, x) = F(s, x)$; and
- $\mathsf{finalize}^F(s) = F(F(0, \mathsf{opad}), s)[1..r]$.

The security of Construction 14 is proved in the F-model, where F is a uniformly random function. To state the theorem for the IT construction, for an integer ℓ, let

$$d'(\ell) = \max_{\ell' \in \{1,\ldots,\ell\}} \left|\{d \in \mathbb{N} \ : \ d|\ell'\}\right| .$$

Observe that, asymptotically, $d'(\ell)$ grows very slowly, i.e., as $\ell^{o(1)}$. Furthermore, let F be a random compression function. The following theorem is equivalent to Lemma 35.

Theorem 50 (IT online extractor from HMAC)**.** *Construction 14 is a $(\gamma^*, q, \varepsilon)$-IT-online extractor in the F-model, where*

$$\varepsilon \leq \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}} + (q+\ell+2) \cdot \frac{2^r}{2^n}} + \rho \quad ,$$

where $\rho = \frac{q^2}{2^n}$.

Proof. We define a few random variables which we will be using in our proofs.

- F a randomly chosen compression function, to which the adversary is given access. We use F both for the oracle itself, as well as for the random variable describing the entire function table.
- ℓ: Number of blocks input to the challenge oracle (which is a random variable itself, we overload notation here, using the same letter we use in the bound on ℓ in the lemma statement).
- $\overline{X} = (\overline{X}_1, \ldots, \overline{X}_\ell)$: the blocks input to the challenge oracle.
- $\tilde{Y}_\ell$: output of HMAC'. Let us remind ourselves that this is s truncated to r bits (the output of $\mathsf{finalize}$);
- $Z = (\Sigma, \mathcal{L}, S_0)$: "side information" where Σ is the attacker state before challenge, $\mathcal{L}$ is the attacker query/answers to F before challenge, S_0 is the initial state provided by $\mathcal{A}$;
- U_r: uniform r-bit string.
- S: the state of the extractor. For the purposes of this proof, we will be using S to indicate the state of the extractor that is the input to $\mathsf{finalize}$.

The advantage of the adversary $\mathcal{A}$ in the online extractor game is bounded by $\mathsf{SD}((\tilde{Y}_\ell, Z, F), (U_r, Z, F))$. Therefore, it is sufficient to upper-bound just that. The reason follows from the fact that Z contains the state of the attacker Σ just before it makes the challenge query. This means that $\mathcal{A}$ cannot tell apart real from random.

Much like the earlier proofs, we define an event $\mathcal{E}$ where the answers to the F-queries by $\mathcal{A}$ are distinct *when truncated to the first r bits*. Note that there are a maximum of q queries in this list. As pointed out earlier $\mathcal{E}$ has the same probability of occurring in either experiment, since the experiments are identical up to the point when this event is defined. Therefore, by Proposition 3,

$$\mathsf{SD}((\tilde{Y}_\ell, Z, F), (U_r, Z, F)) \;\leq\; \mathsf{SD}((\tilde{Y}_\ell, Z, F)|\mathcal{E}, (U_r, Z, F)|\mathcal{E}) + \frac{q^2}{2^r} \;;$$

For convenience we let $\rho := q^2/2^r$ for the remainder of the proof. In order to bound the statistical distance conditioned on $\mathcal{E}$, we can rewrite the same as as

$$\mathsf{SD}((\tilde{Y}_\ell, Z, F)|\mathcal{E}, (U_r, Z, F)|\mathcal{E}) \;=\; \sum_{z \in \mathcal{E}} \mathsf{P}[Z = z|\mathcal{E}] \cdot \mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;, \tag{12}$$

where $z \in \mathcal{E}$ is to denote that the sum is taken over all side informations $Z = z$, satisfying $\mathcal{E}$.[29] Define $p_z := \mathsf{Pred}(\overline{X}|Z = z)$, and observe that

$$\mathbf{E}_z[p_z] \;=\; \mathsf{Pred}(\overline{X}|Z) \;\leq\; 2^{-\gamma^*} \;,$$

where the latter inequality follows from the assumption $\mathrm{H}_\infty(\overline{X}|Z) \geq \gamma^*$. Moreover,

$$\mathrm{H}_\infty(\overline{X}|Z\mathcal{E}) \;\geq\; \gamma^* - \log(1-\rho)^{-1} \;,$$

which is due to

$$\begin{aligned}
\mathsf{Pred}(\overline{X}|Z) \;&\geq\; \sum_{z \in \mathcal{E}} \mathsf{P}[Z = z] \cdot p_z \\
&=\; \mathsf{P}[\mathcal{E}] \cdot \sum_{z \in \mathcal{E}} \frac{\mathsf{P}[Z = z]}{\mathsf{P}[\mathcal{E}]} \cdot p_z \\
&=\; \mathsf{P}[\mathcal{E}] \cdot \mathsf{Pred}(\overline{X}|Z\mathcal{E}) \;.
\end{aligned}$$

From Lemma 51 we prove below, we will get,

$$\mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;\leq\; \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}} + (q + \ell + 2) \cdot \frac{2^r}{2^n}} \;.$$

Using Jensen's inequality, (12) becomes, for $\alpha = 2^r$ and
$\beta = \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}} + (q + \ell + 1) \cdot \frac{2^r}{2^n}$,

$$\begin{aligned}
\mathsf{SD}((\tilde{Y}_\ell, F)|\mathcal{E}, (U_r, F)|\mathcal{E}) &\leq \frac{1}{2}\sqrt{\alpha \mathsf{Pred}(\overline{X}|\mathcal{L}\mathcal{E}) + \beta} \\
&\leq \frac{1}{2}\sqrt{\frac{2^{r-\gamma^*}}{(1-\rho)} + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}} + (q + \ell + 2) \cdot \frac{2^r}{2^n}} \;.
\end{aligned}$$

□

[29] Therefore $\mathcal{E}$ can be omitted in the conditioning of the statistical distance.

Lemma 51. *For $z \in \mathcal{E}$ and the random variables as defined earlier,*

$$\mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;\leq\; \frac{1}{2}\sqrt{2^r p_z + \ell \cdot d'(\ell) \cdot \frac{2^r}{2^n} + 64\ell^4 \cdot \frac{2^r}{2^{2n}} + 16\ell^2 q^2 \cdot \frac{2^r}{2^{2n}} + (q+\ell+2) \cdot \frac{2^r}{2^n}} \;.$$

Proof. Fix $z = (\sigma, L, s_0)$. When we condition on z, it becomes evident that F is uniformly distributed over all functions that agree with L. We now use Proposition 2 to get that:

$$\mathsf{SD}((\tilde{Y}_\ell, F)|z, (U_r, F)|z) \;\leq\; \frac{1}{2}\sqrt{2^r \cdot \mathsf{Coll}(\tilde{Y}_\ell | Fz) - 1} \;. \tag{13}$$

To bound the collision probability, we consider the following experiment:

- choose F uniformly consistent with L
- sample inputs $\overline{X} = (\overline{X}_1, \ldots, \overline{X}_\ell)$ and $\overline{X}' = (\overline{X}'_1, \ldots, \overline{X}'_{\ell'})$ independently but conditioned on $Z = z$.
- compute S, S' as the $\mathsf{refresh}$ evaluations with F of inputs $\overline{X}$ and $\overline{X}'$ respectively.
- compute $\tilde{Y}_\ell$ and $\tilde{Y}'_\ell$ as the truncated HMAC' evaluations with F of $\overline{X}$ and $\overline{X}'$, i.e $\tilde{Y}_\ell = F(F(0, \mathsf{opad}), S)[1..r]$ and $\tilde{Y}'_\ell = F(F(0, \mathsf{opad}), S')[1..r]$

We first condition on the event $\mathcal{E}$ that $F(0, \mathsf{opad}) \neq F(0, \mathsf{ipad})$. Then, we bound the probability that $\tilde{Y}_\ell = \tilde{Y}'_\ell$ in this modified experiment which is conditioned on $\mathcal{E}$ as

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\leq\; \frac{1}{2^n} + \mathsf{P}[\overline{X} = \overline{X}'] + \mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell | \overline{X} \neq \overline{X}'][30] \;. \tag{14}$$

Now, the first term reduces to at most p_z. Towards bounding the second term, we fix arbitrary inputs $\overline{x} \neq \overline{x}'$ of lengths ℓ and ℓ', respectively. We also assume, wlog, that the evaluation of $\overline{x}'$ is not completely covered by L; otherwise it would violate the collision-freeness of L. Let $\overline{x}_{k+1}$ be the first block of $\overline{x}$ not covered by L and similarly $\overline{x}'_{k+1}$ for $\overline{x}_{\prime}$. We let $k = \ell$ if all blocks are covered. We now use the results of Lemma 52 to upperbound $\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell]$. This concludes the proof. □

Lemma 52. *For fixed inputs $\overline{x} \neq \overline{x}'$ and the random variables as defined earlier,*

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\leq\; \frac{\ell \cdot d'(\ell)}{2^n} + \frac{64\ell^4}{2^{2n}} + \frac{16\ell^2 q^2}{2^{2n}} + \frac{q+\ell+1}{2^n} + \frac{1}{2^r} \;.$$

Proof. Upon fixing it to be arbitrary inputs, we can drop the conditioning on $\overline{X} \neq \overline{X}'$. In this setting, we can condition on the equality of S and S' as follows:

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \;\leq\; \mathsf{P}[S = S'] + \mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell | S \neq S'] \;. \tag{15}$$

Let us take a look at the first term. Notice that this is the collision probability of S when run on two fixed inputs $\overline{x}$ and $\overline{x}'$. In other words, this proof is similar to the bounding of collision probability of truncated outputs of MD when run on $\overline{X}$ with side information z. We now proceed to construct the structure graph as done in the Proof of Lemma 36. The key difference here is that we need the structure graph to be colliding, i.e $\mathsf{P}[S = S'] = \mathsf{P}[G_F(\overline{x}, \overline{x}') \in \mathsf{Coll}_L(\overline{x}, \overline{x}')]$ where $\mathsf{Coll}_L(\overline{x}, \overline{x}')$ are the set of colliding structure graphs. We now use the results of Lemmas 38, 39 and 40 to conclude that:

$$\mathsf{P}[S = S'] \leq \frac{\ell \cdot d'(\ell)}{2^n} + \frac{64\ell^4}{2^{2n}} + \frac{16\ell^2 q^2}{2^{2n}} \;. \tag{16}$$

[30] The conditioning on $\mathcal{E}$ is dropped from the latter two terms.

Again, we look at the second term. We can fix the values to be arbitrary $s \neq s'$. Note that conditioning could impact the randomness of the function F. However, when we run on fixed inputs $\overline{x}, \overline{x}'$. This would mean that F is uniformly random conditioned on the set of queries in L and the set of evaluations of $\overline{x}$, $\overline{x}'$ resulting in states s, s' respectively. In this setting, define a bad event as an event $\mathcal{E}$ such that the final output are not the same. Therefore, we can bound:

$$\mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell] \leq \mathsf{P}[\tilde{Y}_\ell = \tilde{Y}'_\ell | \mathcal{E}] + \mathsf{P}[\neg\mathcal{E}] \leq \frac{1}{2^r} + \frac{q+\ell+1}{2^n}$$

where the last inequality is a result of Lemma 37.

We can look at $\mathsf{P}[\neg\mathcal{E}]$ as conditioned on an event that the output of $F(0, \mathsf{opad})$ collides with one of the other at most $q + \ell$ values corresponding to the L and the evaluations of the inputs. When this output does not collide, then the final output is freshly sampled and by the randomness of F we get that the final outputs can collide with probability at most $\frac{1}{2^n}$. Furthermore, the probability that output of $F(0, \mathsf{opad})$ collides is $\frac{q+\ell}{2^n}$. This concludes the proof.

□

C Hybrid Proofs

C.1 Recovering and Preserving Imply Robustness

Theorem 12. *Consider a PRNG construction* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *for which* refresh *makes* α *P-calls and* next *makes* β *P-calls. Furthermore, assume* PRNG *is both*

- $(\gamma^*, q, \ell, \varepsilon_{\text{rec}})$*-recovering and*
- $(q, \ell, \varepsilon_{\text{pre}})$*-preserving*

in the P-model. Then, PRNG *is also* $(\gamma^*, q, t, \ell, \varepsilon_{\text{rob}})$*-robust in the P-model, where*

$$\varepsilon_{\text{rob}} \;\leq\; t \cdot (\varepsilon_{\text{rec}} + \varepsilon_{\text{pre}}) \;.$$

For the proof of Theorem 12, consider a γ^*-legitimate (canonical) (q, t, ℓ)-attacker $\mathcal{A}$. The proof is through a series of hybrid games, where the advantage of $\mathcal{A}$ in the final game is easily seen to be 0. In order to define the hybrids, let a *nice* next query be either **get-next** or **next-ror**. A nice next query is

- *recovering* if it is the first such query after the MRED and
- *preserving* otherwise.

Define the hybrid rob_i to be the robustness game for PRNG where for first i nice next queries, the output of next is replaced by a uniform random string of length $n + r$. Moreover, consider an intermediate hybrid $\text{rob}_{i+\frac{1}{2}}$, where the challenger also replaces the output of next with a random string if the $(i+1)^{\text{st}}$ query is *preserving*.

Denote by η_i the probability that $\mathcal{A}$ outputs 1 when interacting with hybrid rob_i. In the following, preserving security will be used to argue the indistinguishability of hybrids rob_i and $\text{rob}_{i+\frac{1}{2}}$ (Claim 53), and recovering security (Claim 54) for $\text{rob}_{i+\frac{1}{2}}$ and rob_i.

Claim 53. *For all* $i = 0, \ldots, t$, $|\eta_i - \eta_{i+\frac{1}{2}}| \leq \varepsilon_{\text{pre}}$.

Proof. The two hybrids only differ in the case when $(i+1)^{\text{st}}$ next query is *preserving*. Hence, assume that the adversary $\mathcal{A}$ ensures that the $(i+1)^{\text{st}}$ query is indeed preserving, which serves to maximize its advantage. Consider the following attacker $\mathcal{A}'$ against the preserving security of PRNG: Initially, $\mathcal{A}'$ sets

- $b \leftarrow\!\!\$ \{0,1\}$,
- $s \leftarrow 0^n$,
- $j \leftarrow 0$, and
- $\chi \leftarrow \lambda$, where λ is the empty string.

Then, $\mathcal{A}'$ runs $\mathcal{A}$, simulating all oracle calls made by $\mathcal{A}$ answering the queries from $\mathcal{A}$ as follows: At all times, P-queries by $\mathcal{A}$ are simply forwarded by $\mathcal{A}'$ to its own P-oracle and back. Furthermore, while $j \leq i$:

- **adv-refresh**(x): $\mathcal{A}'$ simply ignores the query.
- **set-state**(s'): $\mathcal{A}'$ just sets $s \leftarrow s'$.
- **get-state**: $\mathcal{A}'$ returns s.
- **get-next** or **next-ror**: $\mathcal{A}'$ chooses $(s, y) \leftarrow\!\!\$ \{0,1\}^{(n+r)}$ uniformly at random, increments $j \leftarrow j+1$, and returns y.

Once $j = i+1$, $\mathcal{A}'$ simulates the oracles as follows:

- **adv-refresh**(x): $\mathcal{A}'$ appends x to χ, i.e., $\chi \leftarrow \chi||x$.
- **set-state**(s'): $\mathcal{A}'$ sets $s \leftarrow s'$.
- **get-state**: $\mathcal{A}'$ returns s.
- **get-next**: $\mathcal{A}'$ calls its challenge oracle to obtain $(s, y) \leftarrow$ **chall**(s, χ), increments $j \leftarrow j+1$, and returns y.
- **next-ror**: $\mathcal{A}'$ calls its challenge oracle to obtain $(s, y_0) \leftarrow$ **chall**(s, χ), chooses $y_1 \leftarrow\!\!\$ \{0,1\}^r$, increments $j \leftarrow j+1$, and returns y_b.

Subsequently, i.e., once $j > i+1$, $\mathcal{A}'$ uses the state s returned by **chall** and its P-access to keep simulating the oracles consistent with rob_{i+1}. In the end, $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs.

Let the challenge bit of the challenger for $\mathcal{A}'$ be $\tilde{b}$. Note that when $\tilde{b} = 0$, $\mathcal{A}'$ perfectly simulates hybrid $\text{rob}_{i+\frac{1}{2}}$ for $\mathcal{A}$. Similarly, if $\tilde{b} = 1$, $\mathcal{A}'$ perfectly simulates hybrid rob_{i+1} for $\mathcal{A}$. (Recall that $\mathcal{A}$ is canonical.) □

The next step is to show that the hybrids $\text{rob}_{i+\frac{1}{2}}$ and rob_{i+1} are indistinguishable from each other.

Claim 54. *For all* $i = 0, \ldots, t-1$, $|\eta_{i+\frac{1}{2}} - \eta_{i+1}| \leq \varepsilon_{\text{rec}}$.

Proof. The two hybrids only differ in the case when $(i+1)^{\text{st}}$ next query is *recovering*. Hence, assume that the adversary $\mathcal{A}$ ensures that the $(i+1)^{\text{st}}$ query is indeed recovering, which serves to maximize its advantage. Consider the following attacker $\mathcal{A}'$ against the recovering security of PRNG: Initially, $\mathcal{A}'$ sets

- $b \leftarrow\!\!\$ \{0,1\}$,

- $s \leftarrow 0^n$,
- $j \leftarrow 0$, and
- $\chi \leftarrow \lambda$, where λ is the empty string.

Then, $\mathcal{A}'$ runs $\mathcal{A}$, simulating all oracle calls made by $\mathcal{A}$ answering the queries from $\mathcal{A}$ as follows: At all times, P-queries by $\mathcal{A}$ are simply forwarded by $\mathcal{A}'$ to its own P-oracle and back. Furthermore, while $j \leq i$:

- **adv-refresh**(x): $\mathcal{A}'$ simply ignores the query.
- **set-state**(s'): $\mathcal{A}'$ just sets $s \leftarrow s'$.
- **get-state**: $\mathcal{A}'$ returns s.
- **get-next** or **next-ror**: $\mathcal{A}'$ chooses $(s, y) \leftarrow\!\!\$\ \{0,1\}^{(n+r)}$ uniformly at random, increments $j \leftarrow j + 1$, and returns y.

Once $j = i + 1$, $\mathcal{A}'$ simulates the oracles as follows:

- **adv-refresh**(x): $\mathcal{A}'$ appends x to χ, i.e., $\chi \leftarrow \chi||x$.
- **set-state**(s'): $\mathcal{A}'$ sets $s \leftarrow s'$.
- **get-state**: $\mathcal{A}'$ returns s.
- **get-next**: $\mathcal{A}'$ calls its challenge oracle to obtain $(s, y) \leftarrow$ **chall**(s, χ), increments $j \leftarrow j + 1$, and returns y.
- **next-ror**: $\mathcal{A}'$ calls its challenge oracle to obtain $(s, y_0) \leftarrow$ **chall**(s, χ), chooses $y_1 \leftarrow\!\!\$\ \{0,1\}^r$, increments $j \leftarrow j + 1$, and returns y_b.

Subsequently, i.e., once $j > i + 1$, $\mathcal{A}'$ uses the state s returned by **chall** and its P-access to keep simulating the oracles consistent with rob_{i+1}. In the end, $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs.

Let the challenge bit of the challenger for $\mathcal{A}'$ be $\tilde{b}$. Note that when $\tilde{b} = 0$, $\mathcal{A}'$ perfectly simulates hybrid $\text{rob}_{i+\frac{1}{2}}$ for $\mathcal{A}$. Similarly, if $\tilde{b} = 1$, $\mathcal{A}'$ perfectly simulates hybrid rob_{i+1} for $\mathcal{A}$. (Recall that $\mathcal{A}$ is canonical.)

It remains to argue that $\mathcal{A}'$ is γ^*-legitimate. To that end, recall from Section 4.2 that the legitimacy condition is defined in the *legitimacy* game. Observe in particular, that up to the time $\mathcal{A}'$ makes its challenge queries, $\mathcal{A}$'s view is exactly as it would be in the legitimacy game, in which one considers the following random variables immediately before $\mathcal{A}$ makes the i^{th} call to oracle **get-next** or **next-ror**:

- $\mathcal{L}_i$: the list of P-queries by $\mathcal{A}$ and the corresponding answers;
- Σ_i: the state of $\mathcal{A}$;
- $\overline{X}_i$: vector of inputs provided by $\mathcal{A}$ since the *the most recent entropy drain (MRED)*; and
- S_i: the state of the PRNG immediately after the MRED.

In a similar fashion, consider the following random variables pertaining to $\mathcal{A}'$ just before its call to the challenge oracle:

- X, the input vector $\mathcal{A}'$ provides to **chall**;

- Σ, the state of $\mathcal{A}'$ just before the call to **chall**;
- $\mathcal{L}$, the list of P-queries and answers by $\mathcal{A}'$ before the call to **chall**; and
- S_0; the state $\mathcal{A}'$ provides to **chall**.

That is, it needs to be established that

$$\mathrm{H}_\infty(X|\Sigma\mathcal{L}S_0) \;\geq\; \gamma^* \;. \tag{17}$$

using the argument of the γ^*-legitimacy of $\mathcal{A}$,

$$\mathrm{H}_\infty(\overline{X}_i|\Sigma_i\mathcal{L}_iS_i) \;\geq\; \gamma^* \;.$$

First, it is easily verified that $S = S_i$. Second, observe that Σ only needs to contain, in addition to Σ_i, the values of b and j, which are clearly independent of X. Third, clearly $\mathcal{L} = \mathcal{L}_i$. Finally, note that $X = \overline{X}_i$. From the preceding, (17) follows. □

C.2 IT-Recovering Implies IT-Robustness

Theorem 34. *Let* PRNG = (refresh, next) *be a PRNG for which* refresh *makes* α *P-calls and* next *makes* β *P-calls. Let the* PRNG *be* $(\gamma^*, q, \ell, \varepsilon_{\mathrm{rec}})$*-IT-recovering in the P-model. Then,* PRNG *is also* $(\gamma^*, q, t, \ell, \varepsilon_{\mathrm{rob}})$*-IT-robust in the P-model, where*

$$\varepsilon_{\mathrm{rob}} \;\leq\; t \cdot \varepsilon_{\mathrm{rec}} \;.$$

As seen in the proof of Theorem 12, define the hybrid rob_i to be the IT-robustness game for PRNG where for first i nice next queries, the output of next is replaced by a uniform random string of length $n + r$. We denote by η_i the probability that $\mathcal{A}$ outputs 1 when interacting with hybrid rob_i.

Claim 55. *For all* $i = 0, \ldots, t-1$, $|\eta_i - \eta_{i+1}| \leq \varepsilon_{\mathrm{rec}}$.

The proof of Claim 55 is completely analogous to that of Claim 54 and is therefore omitted.