No Time to Hash:
On Super-Efficient Entropy Accumulation

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

Siyao Guo
New York University Shanghai
siyao.guo@nyu.edu

Noah Stephens-Davidowitz
Cornell University
noahsd@gmail.com

Zhiye Xie
New York University Shanghai
zx572@nyu.edu

Abstract

Real-world random number generators (RNGs) cannot afford to use (slow) cryptographic hashing every time they refresh their state R with a new entropic input X. Instead, they use "superefficient" simple entropy-accumulation procedures, such as

$$R \leftarrow \mathsf{rot}_{\alpha,n}(R) \oplus X,$$

where $\mathsf{rot}_{\alpha,n}$ rotates an n-bit state R by some fixed number α. For example, Microsoft's RNG uses $\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$. Where do these numbers come from? Are they good choices? Should rotation be replaced by a better permutation π of the input bits?

In this work we initiate a rigorous study of these pragmatic questions, by modeling the sequence of successive entropic inputs $X_1, X_2, \ldots$ as *independent* (but otherwise adversarial) samples from some natural distribution family $\mathcal{D}$. Our contribution is as follows.

- We define 2-*monotone distributions* as a rich family $\mathcal{D}$ that includes relevant real-world distributions (Gaussian, exponential, etc.), but avoids trivial impossibility results.
- For any α with $\gcd(\alpha, n) = 1$, we show that rotation accumulates $\Omega(n)$ bits of entropy from n independent samples $X_1, \ldots, X_n$ from any (unknown) 2-monotone distribution with entropy $k > 1$.
- However, we also show some choices of α perform much better than others for a given n. E.g., we show $\alpha = 19$ is one of the best choices for $n = 64$; in contrast, $\alpha = 5$ is good, but generally worse than $\alpha = 7$, for $n = 32$.
- More generally, given a permutation π and $k \geq 1$, we define a simple parameter, the *covering number* $C_{\pi,k}$, and show that it characterizes the number of steps before the rule

$$(R_1, \ldots, R_n) \leftarrow (R_{\pi(1)}, \ldots, R_{\pi(n)}) \oplus X$$

  accumulates nearly n bits of entropy from independent, 2-monotone samples of min-entropy k each.
- We build a simple permutation π^*, which achieves nearly optimal $C_{\pi^*,k} \approx n/k$ for all values of k *simultaneously*, and experimentally validate that it compares favorably with all rotations $\mathsf{rot}_{\alpha,n}$.

1 Introduction

Good random number generation is essential for cryptography and beyond. In practice, this difficult task is solved by a primitive called a *Random Number Generator (RNG, or RNG with input)*, whose aim is to quickly accumulate entropy from various physical entropic sources in the environment with unknown distributions (such as timing of interrupts, etc.). The RNG then converts this high-entropy state into the (pseudo)random bits that are needed for applications. In this work we focus on the first step: *entropy accumulation*. This is usually achieved by a procedure $S \leftarrow \mathsf{Refresh}(S, X)$, where S is the state of the RNG, and X is the entropic input whose entropy we are trying to "accumulate" into the fixed-length RNG state S.[1] Intuitively, we wish to design Refresh so that S converges to a high-entropy, and eventually (almost) uniform distribution, provided that the input samples $X_1, X_2, \ldots$ *collectively* have enough entropy, without too many additional assumptions about the X_i.

In the context of RNGs, the requirement of entropy accumulation was formalized by Dodis et al. [DPR+13] (building on prior influential work of [BH05]), and there has been much follow-up work [DSSW14, GT16, Hut16, CDKT19]. Most of these works consider a very powerful adversary, who tries to choose the *worst* possible entropy source for the Refresh subject to satisfying the overall entropy constraints. As such, all existing Refresh procedures in the literature are relatively expensive, using either a cryptographic hash function Hash which simply sets $S \leftarrow \mathsf{Hash}(S, X)$ for the new input X, or, under some additional assumptions [DPR+13], a full field multiplication over a finite field $\mathsf{GF}[2^N]$ for large values of N (on the order of 500-1000).

Unfortunately, the Refresh procedures from these theoretical works appear to miss the following critical consideration, making them insufficient for real-world RNG design. Many practical entropy sources — such as interrupt timings — come at a very rapid pace (but possibly with relatively low entropy per sample). Hence, running a cryptographically secure hash function (or doing a very large finite field multiplication) every time we receive such an input X would be not only be prohibitively expensive, but *completely infeasible* for an operating system RNG, for example.

As a result, practitioners use the following elegant compromise, not yet modeled by the theory of RNGs (prior to our work), but found in every major operating system including `/dev/random` [Wik04] for Linux, Yarrow [KSF99] for MacOs/iOS/FreeBSD, and Fortuna [FS03] for Windows [Fer13, Fer19]. The state of the RNG will consist of two pieces: a relatively long state S for the "slow" entropy accumulation procedure we denote by Slow-Refresh, and a small array of very short states R — sometimes called *registers* — for the "superefficient" entropy accumulation procedure Fast-Refresh. On every single interrupt timing X, one always updates one of the registers R (usually in some round-robin manner):

$$R \leftarrow \mathsf{Fast\text{-}Refresh}(R, X)$$

Since interrupts could happen very frequently, the mandatory requirement for the fast refresh operation is *extreme speed and simplicity*. We comment on this below, but first complete the refresh procedure description. Less frequently, one would accumulate the state of all the registers $\{R\}$ into the long RNG state S:

$$S \leftarrow \mathsf{Slow\text{-}Refresh}(S, \{R\})$$

[1] Equivalently, one can think of the refresh procedure as a *randomness condenser* [RR99, RSW00], which condenses $|S|+|X|$ bits back to $|S|$ bits, while trying not to lose the overall entropy in *both* S and X (and therefore "accumulating" the fresh entropy brought by X back into the state S).

The latter function is typically implemented as a cryptographic hash function Hash, and can afford to be much slower. It is then this longer state S that will be used to generate (pseudo)random bits. This in particular means that the registers R do not need to achieve the same guarantees as the larger state S.

All existing theoretical modelings of RNGs with input only focused on the slow accumulation procedure Slow-Refresh. As such, it completely abstracted away a key question concerning the design of all practical RNGs:

What is the best way to design extremely fast and practical refresh procedures Fast-Refresh *to accumulate entropy as fast as possible?*

The goal of this work is to model these super-efficient entropy accumulators, and to build the theoretical foundations for this very important primitive. Hence, for much of this work (with the exception of Section 8), we will completely ignore Slow-Refresh (and all other details of RNGs), and focus exclusively on the clear question of understanding the design of super-efficient entropy accumulation.

Existing Designs: Cyclic Rotation. To dig into our question a bit deeper, it is helpful to see what is typically done in practice. As we said, the fast-refresh procedure has to be blazing fast, and can realistically involve just a few simple bit-level operations applied to the entropic input and the register. In fact, most RNGs we know, such as the one used by Windows 10 [Fer19], implement the following "rotate-then-xor" procedure. The register R is typically an $n = 32$ or $n = 64$-bit value. The raw input X—such as the timing of the previous interrupt—is also an n-bit string. To refresh the register (quickly!), one simply cyclically rotates the bits of the register by some fixed constant α (e.g., rotation by two would map the bit string $(1, 1, 0, 1, 0, 1)$ to $(0, 1, 1, 1, 0, 1)$), and then XORs the input X to the result:

$$R \leftarrow \mathsf{rot}_{\alpha,n}(R) \oplus X$$

Concretely, Microsoft uses $\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$. [Fer19].

Our Questions. While this design appears reasonable, it raises a lot of questions that we would like to answer.

- How were these (seemingly mysterious) rotation numbers α selected?
- Is there some rigorous metric/model that can help compare different rotation amounts to each other, either practically, or theoretically, or both?
- In particular, are Microsoft's choices of $\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$ "good"?
- How should one model the distributions of the entropic inputs X to properly study these refresh procedures?
- Is rotation really the best way to permute the bits of the state for entropy accumulation?
- In particular, can rotation be replaced by a "better" permutation π of the n register positions: $(R_1, \ldots, R_n) \leftarrow (R_{\pi(1)}, \ldots, R_{\pi(n)}) \oplus X$?

To start answering these questions informally, let us make some simple observations to get some intuition for why Microsoft might have chosen these seemingly mysterious numbers, 5 and 19. First, it seems clear that we should take the rotation amount α relatively prime to n, to make sure every bit eventually affects every other bit. Second, we claim that it is unwise to take α very small (e.g., $\alpha = 1$), since practical sources will tend to "have most of their entropy in the lower-order bits," so that small values of α will take a lot of time to affect all the bits of the register. For example, even if every sample of X is uniform in its $n/2$ least significant bits, rotation by 1 will only accumulate $n/2 + \ell - 1$ bits after ℓ steps. For a similar reason, one should avoid values of α where a *small multiple* of α is very close to n; such as $\alpha = 11$ for $n = 32$, or $\alpha = 21$ for $n = 64$. For example, after three such steps with $\alpha = 11$ for $n = 32$, a fresh sample which is uniform in its 5 least-significant bits will contribute only one fresh bit of entropy, just as if we had $\alpha = 1$.

Choosing between the remaining possibilities of, e.g., $\alpha = 5, 7, 9, \ldots$, yields subtle tradeoffs. Indeed, while it is clear that there is something interesting going on here, it is not immediately clear how to formalize this.

1.1 Our model

In this work, we use the tools of modern information theory and cryptography to make the above ad-hoc arguments more systematic and theoretically sound, so that we have higher confidence in the quality of our answers. In the process, we will uncover some interesting theory.

Syntax and Efficiency. First, we restrict our attention to entropy accumulation of the form

$$(R_1, \ldots, R_n) \leftarrow (R_{\pi(1)}, \ldots, R_{\pi(n)}) \oplus X$$

for some permutation $\pi : [n] \to [n]$ of the n-bit register R, as this model is quite natural in our context of super-efficient constructions. For conciseness, we let $A_\pi(R) = (R_{\pi(1)}, \ldots, R_{\pi(n)})$, with cyclic rotation $\mathsf{rot}_{\alpha,n}(R)$ being of most immediate interest to us.

Modeling of Entropic Inputs X. Given the extreme simplicity of our accumulation procedure, it is clear that we will not be able to withstand the same level of generality and "malicious" attacks that are modeled in prior work addressing the complementary question of "slow refresh". For example, even if the marginal distributions of X_i are completely uniform in $\{0,1\}^n$, we will fail to accumulate a single bit of entropy if, for example, the X_i satisfy $A_\pi(X_{2i-1}) = X_{2i}$. (The state will be zero after every even number of steps when starting from $R = 0^n$!)

Hence, as the first modeling assumption we will assume that the inputs X_i are *independent*. This is a common abstraction in the randomness extraction literature dating back to [CG88].[2] While it might not be entirely accurate in practice, we believe that it captures some of what is useful about natural sources such as interrupt timings, which do not appear fully adversarial.

Second, to minimize the number of parameters, and also to focus on the high-level picture, in our analyses we will assume that the entropy of each (independent) sample is lower bounded by some parameter k. (Once again, this is standard in the randomness extraction literature; with very few exceptions, such as [KRVZ11].) The key point is that our refresh procedure does not know/use anything about k, and a "good" result should yield quick entropy accumulation for *all values* of k; presumably in roughly n/k steps, which is the best possible. Thus, even if the quality of source is

[2]See also [BIW04, KRVZ11, CZ19] for some exciting advances in the area of randomness extraction from independent sources.

unknown, a "good" result of this type will tell us that our entropy accumulation always works to the best extent possible.

Finally, we will further restrict each sample to come from some (natural) family of distributions $\mathcal{D}$. (This is also common in the literature. E.g., [BTRS02] did so in the context of slow refresh.) Indeed, it is easy to see that our refresh procedure is too simple to work for arbitrary (even independent) distributions of entropy k. For example, if only k bits of X_0 have entropy, it is trivial to see where these k bits "travel" after i mixing steps given by π. Say, for rotation by 1, after i such rotations the first k bits $(1, \ldots, k)$ go to locations $(1 + i \bmod n, \ldots, k + i \bmod n)$. Thus, in this example (which is easy to generalize to any π) one can define X_i to be uniform over positions $(1+i \bmod n, \ldots, i+k \bmod n)$. This gives k independent bits of entropy, but this entropy is repeatedly added to the same place (just shifted over and over). Hence, we can never accumulate any entropy beyond the first k bits in this example.

Two-Monotone Distributions. Of course, the example above seems rather artificial, and unlikely to occur in the actual distributions encountered by these RNGs. (E.g., it seems implausible that the distribution of interrupt timings could have, say, the 10th bit uniformly random but the least significant bit fixed.) Thus, we must choose an appropriate family of distributions. We need *some* restriction on our sources to avoid the counterexamples above, but we would of course like to work with the most general class of distributions possible.

As our first main contribution, we provide a definition that is quite general but sufficient for our purposes. Indeed, as we will discuss more below, for this class of distributions, we are able to formalize the intuitive requirement that "natural distributions have most of their entropy in the lower-order bits."

Specifically, we define a very wide class of distribution, which we call 2-*monotone*. These are n-bit distributions such that the probability mass function over $\{0, \ldots, 2^n - 1\}$ (i.e., interpreting the n bits as an integer written in binary) "has at most one peak." (Formally, the distribution is 2-monotone if we can divide $\mathbb{Z}_{2^n}$ into two intervals such that the probability mass function is monotone on the two intervals. See Section 3.) This is a large class, and it includes, e.g., Gaussians over $\mathbb{Z}_{2^n}$, exponential distributions over $\mathbb{Z}_{2^n}$, and uniform distributions over an interval — three natural distributions that one might use to model, e.g., the timing of interrupts.

We then show that any such distribution does in fact "have most of its entropy in the lower order bits." (The precise statement is Fourier analytic. See Lemma 3.2.) This will help us overcome the impossibility result sketched above, while maintaining a (surprisingly!) large level of generality. To summarize, we will instantiate our family of distributions to be $\mathcal{D}_{k,n}$ — all two-monotone distributions on n bits having entropy at least k, and will allow arbitrary independent (but *not necessarily identical*) choices of entropic inputs $X_1, X_2, \ldots \in \mathcal{D}_{k,n}$.

Goal: Entropy Accumulation. We must also select the notion of entropy for the register R for our goal of entropy accumulation. As our default choice, we will use the standard notion of min-entropy, $H_{\min}(R)$. This is a conservative notion of entropy which is enough to be composed with any Slow-Refresh procedure (or any other randomness extractor [NZ96]) from the literature. However, some RNGs [DPR+13, CDKT19],[3] and all randomness extractors based on the famous leftover hash lemma [HILL99], can work for a less conservative notion of entropy, called *collision entropy* $H_2(R)$. Hence, in our results we will keep track of both the min- and the collision entropy of

[3]This is not stated in the results of [DPR+13, CDKT19], but is implicit from the technical analysis.

R.[4] Indeed, our collision entropy results will be, as expected, slightly better than the min-entropy bounds.

Putting everything together, we arrive at the following clean question:

Main Question: *For given n, k, permutation π, and number of iterations ℓ, what is the min-/collision entropy of R_ℓ, where $R_0 = 0^n$, $R_i = A_\pi R_{i-1} \oplus X_i$, and $X_1, X_2, \ldots, X_\ell$ are independent two-monotone distributions from $\mathcal{D}_{k,n}$?*

Bigger Picture. We stress once again that our question is largely complementary and incomparable to the analyses of "slow refresh" procedures from all the prior work [DPR+13, DSSW14, GT16, Hut16, CDKT19]. Slow refresh operates on much larger block size $N \gg n$, is concerned with randomness extraction rather than accumulation, and attempts to defend against much more powerful attacks. In order to achieve this, the slow-refresh procedure must necessarily be much slower than our fast-refresh procedure. In particular, the two procedures are used in different, complementary places in the overall RNG design: the array or registers becomes an input to the slow-refresh procedure after many fast-refresh calls. In Section 8, we briefly discuss how our results might start filling the "missing link" in the prior RNG work, but stress once again that they cannot be meaningfully compared to each other.

1.2 Our contributions

Rotation performs reasonably well. Recall that we show a key property of 2-monotone distributions: they "have most of its entropy in the lower order bits." (The precise statement is Fourier analytic. See Lemma 3.2.) Using this characterization, we can then relatively easily show that any rotation on n bits (with α coprime to n, or, indeed, any cyclic permutation) can accumulate nearly a full n bits of entropy in n steps.

Theorem 1.1 (Informal, see Theorem 4.1)**.** *Any rotation on n bits (with rotation number α coprime to n) will accumulate (approximately) $n(1 - 2^{-2k+2})$ bits of collision entropy and (approximately) $n(1 - 2^{-k+1})$ bits of min-entropy from any n independent sources in $\mathcal{D}_{k,n}$, for $k > 1$.*

Comparing different rotations using covering number. Theorem 1.1 justifies the use of rotation, but only if we are willing to wait n steps (regardless of how large k is) and fails to distinguish between different rotation numbers α. Indeed, as we discussed above, when $\alpha = 1$, we do in fact need roughly n steps in order to accumulate nearly n bits of entropy, even if the input already has very high entropy. So, if we wish to do better, we must somehow distinguish between different rotation numbers.[5]

To do this, we introduce a simple, efficiently computable quantity $C_{\alpha,k}$, which we call the *covering number*. Intuitively, $C_{\alpha,k}$ is the number of steps needed for rotation by α to accumulate

[4]Our results will eventually give standard randomness extractors, when R accumulates nearly a full n bits of entropy (see Appendix C). However, we choose to focus on entropy accumulation, as (1) this is the use of superefficient entropy accumulators in real-world applications; (2) the restrictive format of our accumulators—while sufficient to quickly get to nearly n bits of entropy (which is our goal!)—will be wasteful in "squeezing the last few bits" of entropy needed for extraction.

[5]It is easy to see that all rotations perform identically if we wait exactly n steps. So, this question is essentially only interesting for fewer than n steps.

full entropy when the input is uniform on $\{0, \ldots, 2^k - 1\}$. Equivalently, $C_{\alpha,k}$ is the minimal number m such that $\{i + \alpha j \bmod n \ : \ 0 \le i < k,\ 0 \le j < m\} = [n]$, i.e., the minimal m such that "m rotations of the first k bits are sufficient to cover all bits." It is easy to see that the covering number is at least n/k and at most $n - k + 1$.

Notice that the covering number is exactly the number of steps needed to accumulate full entropy from the (two-monotone) distribution in which the first k bits are uniform and independent, while the remaining $n - k$ bits are fixed. We show (using Fourier-analytic techniques) that the covering number actually characterizes the performance of rotation by α on *all* 2-monotone distributions with entropy k, up to a factor of 2 in k. In other words, up to this factor of 2 in k (and ignoring the specific notion of "accumulating enough entropy"), the uniform distribution on $\{0, \ldots, 2^k - 1\}$ is "the worst case".

Theorem 1.2 (Informal, see Theorem 5.2)**.** *Let $m := C_{\alpha,\lfloor k/2 \rfloor}$ and $k \ge 2$. After m steps, rotation by α accumulates at least $n \cdot (1 - 2^{-k})$ bits of collision entropy and $n \cdot (1 - 2^{-k/2})$ bits of min-entropy from any distribution in $\mathcal{D}_{k,n}$. Alternatively, it accumulates at least $n - 1$ bits of collision entropy after $m(1 + \log(n/k)/k)$ steps, and $n - 1$ bits of min-entropy after $m(1 + 2\log(n/k)/k)$ steps.*

Theorem 1.2 suggests comparing rotations according to their covering numbers $C_{\alpha,k}$, effectively reducing a seemingly very difficult problem to a simple calculation. Therefore, we compute these covering numbers for different rotations. While there is no unambiguous ranking of the different rotations,[6] we show that some rotations perform well in general, while others do not. (E.g., $C_{11,k} > n - 3k$ for $n = 32$.) In particular, Microsoft's choice of $\alpha = 19$ when $n = 64$ is quite reasonable (though $\alpha \in \{15, 23, 27\}$ also seem like reasonable choices). Microsoft's choice of $\alpha = 5$ for $n = 32$ is also reasonable, though we observe that certain other choices, particularly $\alpha = 7$ and $\alpha = 9$, also perform reasonably well for all k and perform noticeably better when the input has high entropy. See Figures 3 and 4 for the data. (See [sup] for a table with all covering numbers for $n = 32$ and $n = 64$.)

Other Permutations and Tightness. Our analysis of the covering number above extends immediately to any cyclic permutation $\pi : [n] \to [n]$. Specifically, the covering number $C_{\pi,k}$ of π essentially characterizes its behavior as an entropy accumulator when its input is a 2-monotone source with entropy k (up to a factor of 2 in k). In fact, in Theorem 5.3 we show that this generalization of Theorem 1.2 is quite tight. In particular, there exists a distribution $D \in \mathcal{D}_{k,n}$ (in fact, the same distribution that we use for our empirical results discussed below) such that no permutation π (including all rotations and the new permutation we discuss below) accumulates more than $n - 1$ bits of collision entropy from D in fewer than $n\log(n)/(k^2 + 4)$ steps. Similarly, it takes at least $2n\log(n)/(k^2 + 4)$ steps to accumulate $n - 1$ bits of min-entropy from this distribution.

Notice, in particular, that our upper and lower bounds nearly match when one sets $m \approx n/k$. (While $m = C_{\pi,\lfloor k/2 \rfloor}$ cannot be smaller than $2n/k$, as we describe below in more detail, we expect that this factor of two is an artifact of our proof and that taking $m \approx C_{\pi,k}$ is a good heuristic. Since $C_{\pi,k}$ can be as small as $\lceil n/k \rceil$, this suggests taking $m \approx n/k$.)

A different permutation: bit-reversed rotation. Our characterization of condensing in terms of $C_{\pi,k}$ motivates us to find a permutation π whose covering number $C_{\pi,k}$ is small for all k; ideally, $C_{\pi,k} \approx n/k$, which is the minimal possible covering number. Indeed, in this regime, our condensing

[6]Some rotations will perform very well for some k, and others will perform well for other k. E.g., for $\alpha = k$, $C_{\alpha,k} = \lceil n/k \rceil$ is always minimal. So every rotation has an optimal covering number for at least one choice of k.

is provably the best possible: we accumulate almost all k bits of input entropy for each of the first nearly n/k steps.

To that end, we construct a permutation that we call *bit-reversed rotation*. This is the permutation obtained by (1) associating the ith bit with the $(\log_2 n + 1)$-bit string i written in binary; (2) setting $\sigma(i)$ to be the number obtained by *reversing* this bit string; and (3) sending the ith bit to the unique position j with $\sigma(j) + 1 = \sigma(i) \bmod 2^n$. See Figure 14 for an illustration. This permutation actually arises naturally from a simple greedy construction in this context,[7] and it satisfies $C_{\pi,k} = n/k$ whenever n and k are both powers of two. I.e., it has optimal covering number $C_{\pi,k}$ for all powers of two k simultaneously! (For general k, the covering number is always bounded by $2n/k$; see Theorem 6.2.)

In Figure 5, we compare covering numbers of bit-reversed rotation against covering numbers of rotation-by-5 for $n = 32$ and rotation-by-19 for $n = 64$ used by Microsoft (and the optimal value n/k). We see that bit-reversed rotation seems to perform at least as well as rotation, and better in several regions. Thus, while we leave it to practitioners to determine whether implementing our new permutation would be preferable in the context of their RNGs, our study suggests that it seems to be the most natural choice from a theoretical perspective. (More on this in our experimental results below.)

Experimental results to compute the exact number of samples needed. Theorem 1.2 (and its generalization in Theorem 5.2) gives strong theoretical justification for using a cyclic permutation with low covering numbers. However, this loss of a factor of 2 in k (i.e., the fact that the theorem requires $C_{\pi,\lfloor k/2 \rfloor}$ samples instead of $C_{\pi,k}$ steps) is unfortunate—especially for the practical case that interests us most, in which n is a small constant like $n = 32$ or $n = 64$. For practical applications, we care about the fine-grained detail of the performance, and we expect that $C_{\pi,k}$ is in fact the right answer, as in the following heuristic.

We expect that (just slightly more than) $C_{\pi,k}$ steps should be sufficient to accumulate nearly full entropy from 2-monotone sources with entropy at least k.

This is of course true—essentially by definition—for the special case of the uniform distribution over $\{0, \ldots, 2^k - 1\}$, and also gives us a lower bound for the general class $\mathcal{D}_{k,n}$.

To that end, in Theorem 7.2 we use the Fourier-analytic theoretical machinery that we used to prove Theorem 1.2 in order to derive a *closed form* expression for the exact (min- or collision) entropy accumulated by any permutation when the input source is an *exponential* distribution. (In other words, the distribution in which the probability that an interrupt happens at time t is proportional to $e^{-t/\sigma}$ for some $\sigma > 0$.) This is a natural example of a 2-monotone distribution (and far less trivial than the uniform distribution), and this closed form lets us compute exactly the number of samples needed to accumulate, say, $(n - 1)$ bits of min-/collision entropy.[8]

These *exact* calculations allow us to answer three interesting questions, at least for the clean and natural case of exponential distributions:

1. Exactly how close is $C_{\pi,k}$ to the *actual* number of samples to accumulate nearly n (say, $(n-1)$) bits of entropy close?

[7]It also arises naturally in other contexts, such as in the fast Fourier transform (in the form of the bit-reversed involution, which we call σ above).

[8]As we see from both our theoretical and our experimental results, our accumulators quickly collect almost n bits of entropy, at nearly optimal pace of k bits per sample, but squeezing the last couple of bits (i.e., becoming an extractor) takes many more samples. This is why we stop our experiments at $(n - 1)$ bits of entropy.

2. Does bit-reversed rotation perform at least as well as any rotation by α?

3. How much faster does collision entropy accumulate compared to min-entropy?

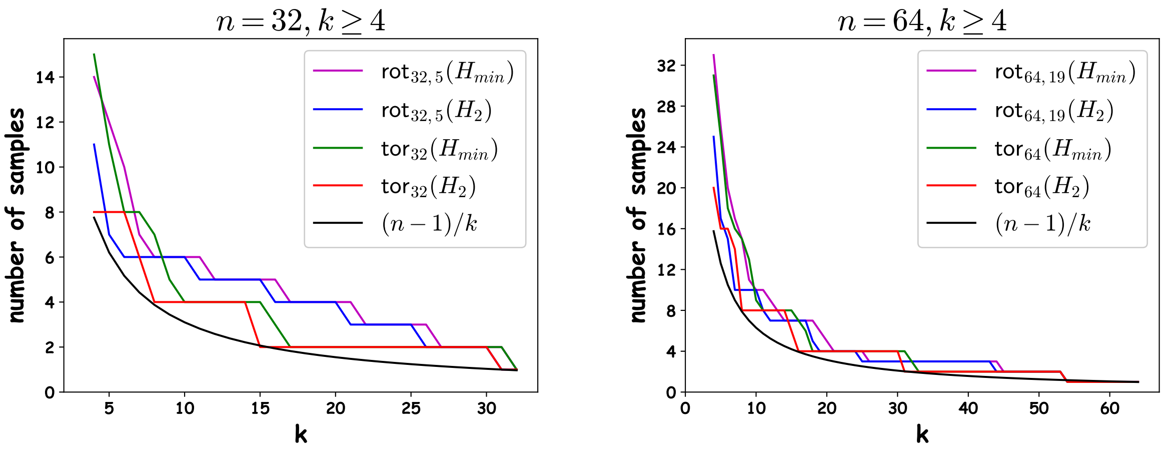

Figure 1: Comparison between the exact number of samples needed to condense to 31 bits of collision/min- entropy (or 63 bits) from the exponential distribution, for bit-reversed rotation and the rotations used by Microsoft with input entropy k.

Our empirical results show that (at least for this natural distribution), (1) the true number of samples needed to accumulate from k bits of entropy to nearly n bits of entropy is very close to $C_{\pi,k}$; (2) bit-reversed rotation compares quite favorably with rotation by α; and (3) collision entropy accumulates slightly but notably faster than min-entropy. There are some subtleties, though. (See Figure 1 for the high-level picture.)

First, as we see in Figure 6, for the case of collision entropy the number of samples is nearly exactly $C_{\pi,k}$, verifying for the intuition that $C_{\pi,k}$ (rather than $C_{\pi,k/2}$ or something in between) controls the number of steps needed to accumulate nearly full entropy. In fact, the results are quite striking. The results for min-entropy in Figure 7 are less striking, especially for low values of k, but still indicate that $C_{\pi,k}$ is the right parameter to look at.

Second, although there is no strict dominance, it is clear that bit-reversed rotation compares favorably with all rotation-by-α results, including Microsoft's choices $\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$. This holds for both min- and collision entropy. See Figure 1 (or, for a more detailed comparison, Figures 8 and 9). Combined with its theoretical motivation, this suggests that bit-reversed rotation might be a better permutation choice than rotation for superfast entropy accumulation, at least pending the question (see Section 6.1) of whether it can be implemented efficiently enough to be used inside the RNG.

Finally, our exact results in Figure 1 (and the more detailed comparison in Figures 10 and 11) also confirm the intuition that collision entropy accumulates notably faster than min-entropy. While this gap is much less than the worst case factor-of-two gap between the two notions of entropy (see Appendix C), it could be noticeable enough to have a significant impact on applications where collision entropy is enough, such as those that rely on the leftover hash lemma [HILL99].

Summary. Overall, we believe that our work provides both theoretical and practical results to shed light on a previously unexplored, but significant aspect of all practical RNGs: the design of "superefficient" entropy accumulation functions.

2 Preliminaries

For an integer $n \geq 1$, we write $[n] := \{0, \ldots, n-1\}$. For a distribution D over $\{0,1\}^n$ and $\boldsymbol{x} \in \{0,1\}^n$, we write $D(\boldsymbol{x}) := \Pr_{\boldsymbol{X} \sim D}[\boldsymbol{X} = \boldsymbol{x}]$ for the probability that D assigns to $\boldsymbol{x}$.

The *min-entropy* and *collision entropy* of D are

$$H_{\min}(D) := \min_{x \in \{0,1\}^n} \log_2(1/D(x)) \text{ and } H_2(D) := \log_2(1/\sum_x D(x)^2) .$$

We will consider the problem of converting independent samples from a distribution D with some min-entropy into a new distribution with large min-/collision entropy. For completeness, in Appendix C we recall the known relationships between H_2 and $H_{\min}$ (as well as other common measures, such as the statistical distance from the uniform distribution and the smoothed min-entropy).

For a distribution D over $\{0,1\}^n$ and $\boldsymbol{w} \in \{0,1\}^n$, we define the Fourier coefficient of D at $\boldsymbol{w}$ as

$$\widehat{D}(\boldsymbol{w}) := \mathop{\mathbb{E}}_{\boldsymbol{X} \sim D}[(-1)^{\langle \boldsymbol{X}, \boldsymbol{w} \rangle}] = \Pr_{\boldsymbol{X} \sim D}[\langle \boldsymbol{X}, \boldsymbol{w} \rangle = 0 \bmod 2] - \Pr_{\boldsymbol{X} \sim D}[\langle \boldsymbol{X}, \boldsymbol{w} \rangle = 1 \bmod 2] .$$

Fact 2.1. *For any distribution D over $\{0,1\}^n$, and $\boldsymbol{x}$ in $\{0,1\}^n$,*

$$D(\boldsymbol{x}) = \frac{1}{2^n} \sum_{\boldsymbol{w} \in \{0,1\}^n} \widehat{D}(\boldsymbol{w})(-1)^{\langle \boldsymbol{x}, \boldsymbol{w} \rangle} .$$

Theorem 2.2 (Parseval's theorem)**.** *For any distribution D over $\{0,1\}^n$,*

$$\sum_{\boldsymbol{x} \in \{0,1\}^n} D(\boldsymbol{x})^2 = 2^{-n} \sum_{\boldsymbol{w} \in \{0,1\}^n} \widehat{D}(\boldsymbol{w})^2 .$$

Corollary 2.3. *For any distribution D over $\{0,1\}^n$,*

$$H_2(D) = n - \log_2 \Big(\sum_{\boldsymbol{w} \in \{0,1\}^n} \widehat{D}(\boldsymbol{w})^2 \Big) ,$$

and

$$H_{\min}(D) \geq n - \log_2 \Big(\sum_{\boldsymbol{w} \in \{0,1\}^n} |\widehat{D}(\boldsymbol{w})| \Big) .$$

Corollary 2.3 shows that the sum of the squares of Fourier coefficients characterizes the collision entropy, and the sum of the absolute values of Fourier coefficients is useful for bounding min-entropy.

Proof. By Parseval's theorem, we have

$$H_2(D) = \log_2(1/\sum_x D^2(x)) = \log_2(2^n / \sum_{\boldsymbol{w} \in \{0,1\}^n} \widehat{D}^2(\boldsymbol{w})) ,$$

which implies the desired conclusion. By Fact 2.1,

$$\begin{aligned}
H_{\min}(D) &= \min_{x\in\{0,1\}^n} \log_2(1/D(x)) \\
&= \min_{x\in\{0,1\}^n} \log_2(2^n/\sum_{\boldsymbol{w}\in\{0,1\}^n} \widehat{D}(\boldsymbol{w})(-1)^{\langle \boldsymbol{x},\boldsymbol{w}\rangle}) \\
&\geq \log_2(2^n/\sum_{\boldsymbol{w}\in\{0,1\}^n} |\widehat{D}(\boldsymbol{w})|)
\end{aligned}$$

as desired. □

The Fourier coefficients arise naturally in our context because they interact nicely with both convolution and linear transformations, as this next well-known claim shows.

Claim 2.4. *For distributions $D_1,\ldots,D_m$ over $\{0,1\}^n$ and linear transformations $A_1,\ldots,A_m \in \mathbb{F}_2^{n\times n}$, let D be the distribution given by*

$$\Pr_{\boldsymbol{X}\sim D}[\boldsymbol{X}=\boldsymbol{x}] = \Pr_{\boldsymbol{X}_1\sim D_1,\ldots,\boldsymbol{X}_m\sim D_m}[A_1\boldsymbol{X}_1 \oplus \cdots \oplus A_m\boldsymbol{X}_m = \boldsymbol{x}] \ ,$$

where the $\boldsymbol{X}_i$ are independent. Then,

$$\widehat{D}(\boldsymbol{w}) = \widehat{D_1}(A_1^T\boldsymbol{w})\cdots\widehat{D_m}(A_m^T\boldsymbol{w}) \ .$$

for any $\boldsymbol{w}\in\{0,1\}^n$.

Proof. We have

$$\begin{aligned}
\mathbb{E}[(-1)^{\langle \boldsymbol{w},\boldsymbol{X}\rangle}] &= \mathbb{E}[(-1)^{\langle \boldsymbol{w},A_1\boldsymbol{X}_1\oplus\cdots\oplus A_m\boldsymbol{X}_m\rangle}] \\
&= \mathbb{E}[(-1)^{\langle \boldsymbol{w},A_1\boldsymbol{X}_1\rangle}]\cdots\mathbb{E}[(-1)^{\langle \boldsymbol{w},A_m\boldsymbol{X}_m\rangle}] \\
&= \mathbb{E}[(-1)^{\langle A_1^T\boldsymbol{w},\boldsymbol{X}_1\rangle}]\cdots\mathbb{E}[(-1)^{\langle A_m^T\boldsymbol{w},\boldsymbol{X}_m\rangle}] \\
&= \widehat{D_1}(A_1^T\boldsymbol{w})\cdots\widehat{D_m}(A_m^T\boldsymbol{w}) \ .
\end{aligned}$$

□

For a distribution D over $\{0,1\}^n$, integer $\ell \geq 1$, and linear transformation $A : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we write $D_A^{(\ell)}$ for the distribution obtained by sampling $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_\ell$ independently and returning $\boldsymbol{X}_1 \oplus A\boldsymbol{X}_2 \oplus \cdots \oplus A^{\ell-1}\boldsymbol{X}_\ell$.

3 Capturing natural distributions

In this section, we consider natural distributions over the integers (e.g., the kinds of distributions that one might expect for interrupt timings). We associate with each integer $0 \leq x < 2^n$ the vector $\boldsymbol{x} = (x_0,\ldots,x_{n-1}) \in \{0,1\}^n$ given by its binary representation. In other words, the $x_i \in \{0,1\}$ are the unique bits that satisfy $x = \sum 2^i x_i$. For example, x might correspond to the timing of a keystroke.

We observe that many natural distributions are captured by the general class of 2-*monotone* distributions, which we define below. See Section 7 for examples of natural distributions that are 2-monotone.

Definition 3.1 (2-monotone distributions over $\mathbb{Z}_{2^n}$)**.** *A function $p : [2^n] \to [0,1]$ is monotone over an interval $\{i_1, i_1+1, \ldots, i_2\}$ if*

$$p[i_1 \bmod 2^n] \le \cdots \le p[i_2 \bmod 2^n] \ \text{ or } \ p[i_1 \bmod 2^n] \ge \cdots \ge p[i_2 \bmod 2^n] \ .$$

We say that p is 2-monotone over $\mathbb{Z}_{2^n}$, if there exist $0 \le i_1 < i_2 \le 2^n - 1$ such that p is monotone on the interval $\{i_1, \ldots, i_2\}$ and on the interval $\{i_2, \ldots, 2^n - 1, 2^n, \ldots, 2^n + i_1 - 1\}$.

We say that a distribution D over $\{0,1\}^n$ is 2-monotone over $\mathbb{Z}_{2^n}$ if it is obtained by sampling an integer $0 \le X \le 2^n - 1$ (interpreted as a bit string as above) according to a 2-monotone probability mass function.

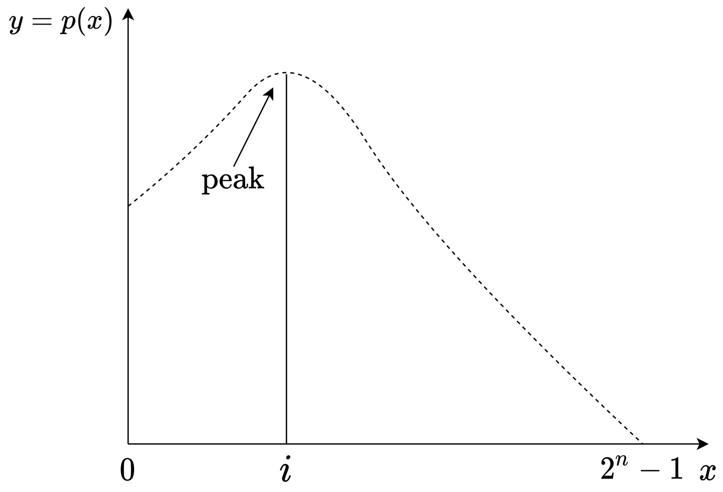

Figure 2: A depiction of a 2-monotone distribution over $\mathbb{Z}_{2^n}$

Intuitively, 2-monotone distributions “change direction at most twice” when viewed as functions on the cycle $\mathbb{Z}_{2^n}$, so that they have “at most one peak” (and “at most one trough”). (For example, all unimodal distributions are 2-monotone.)

A very nice feature of 2-monotone distributions D is that $|\widehat{D}(\boldsymbol{w})|$ is small if $w_i = 1$ for some small index i. This formally captures the intuition that the lower-order bits of “natural distributions” should have high entropy.

Lemma 3.2. *For any 2-monotone distribution D over $\{0,1\}^n$ with min-entropy k, and $\boldsymbol{w} \in \{0,1\}^n$ with $w_i = 1$,*

$$\left|\widehat{D}(\boldsymbol{w})\right| \le \min\{1, 2^{i+1-k}\} \ .$$

The lemma follows immediately from the following two claims.

Claim 3.3. *If $\sum_{j=1}^{2^n-1} |D(j) - D(j-1)| \le \varepsilon$, then for any $\boldsymbol{w}$ with $w_i = 1$,*

$$|\widehat{D}(\boldsymbol{w})| \le \min\{1, 2^i \cdot \varepsilon\} \ .$$

Proof. We have

$$\begin{aligned}
|\widehat{D}(\boldsymbol{w})| &= \Big| \mathop{\mathbb{E}}_{\boldsymbol{X}\sim D}[(-1)^{\langle \boldsymbol{X},\boldsymbol{w}\rangle}] \Big| \\
&= \Big| \sum_{\boldsymbol{X}:\boldsymbol{X}_i=0} (-1)^{\langle \boldsymbol{X},\boldsymbol{w}\rangle}(D(\boldsymbol{X}) - D(\boldsymbol{X}+\boldsymbol{e}_i)) \Big| \\
&\le \sum_{\boldsymbol{X}:\boldsymbol{X}_i=0} |D(\boldsymbol{X}) - D(\boldsymbol{X}+\boldsymbol{e}_i)| \\
&= \sum_{\boldsymbol{X}:\boldsymbol{X}_i=0} |D(X) - D(X+2^i)| \\
&\le \sum_{\boldsymbol{X}:\boldsymbol{X}_i=0} \sum_{j=1}^{2^i} |D(X+j) - D(X+j-1)| \\
&\le 2^i \cdot \sum_{j=1}^{2^n-1} |D(j) - D(j-1)|.
\end{aligned}$$

□

Claim 3.4. *If D is 2-monotone over $\mathbb{Z}_{2^n}$ with min-entropy k, then*

$$\sum_{j=1}^{2^n-1} |D(j) - D(j-1)| \le 2^{1-k} .$$

Proof. Suppose p is monotone on $\{0,\dots,i\}$ and $\{i,\dots,2^n-1\}$ for $0 < i < 2^n-1$.

$$\begin{aligned}
\sum_{j=1}^{2^n-1} |D(j) - D(j-1)| &= \sum_{j=1}^{i} |D(j) - D(j-1)| + \sum_{j=i}^{2^n-1} |D(j) - D(j-1)| \\
&= |D(i) - D(0)| + |D(2^{n-1}) - D(i)| \\
&\le 2^{1-k}
\end{aligned}$$

where the second inequality is by monotonicity of p, and the last inequality is because D has min-entropy k, so $|D(x) - D(y)| \le 2^{-k}$ for every $0 \le x, y \le 2^n-1$. Similarly, if p is monotone on $\{i_1,\dots,i_2\}$ and $\{i_2,\dots,2^n-1,0,\dots,i_1\}$ for $0 < i_1 < i_2 < 2^n-1$, we have

$$\sum_{j=1}^{2^n-1} |D(j) - D(j-1)| \le 2|D(i_1) - D(i_2)| \le 2^{1-k}.$$

The desired conclusion follows. □

4 Rotation condensers

In this section, we set out to understand the power of rotation condensers generically. For integers $0 < \alpha < n$, we write $\mathsf{rot}_{\alpha,n}$ for the linear transformation over $\{0,1\}^n$ defined by

$$\mathsf{rot}_{\alpha,n}((x_1,\dots,x_n)) := (x_{1+\alpha}, x_{2+\alpha}, \dots, x_n, x_1, \dots, x_\alpha) .$$

I.e., $\mathsf{rot}_{\alpha,n}$ rotates the coordinates of a vector $\boldsymbol{x}$ by α. Notice that $\mathsf{rot}_{\alpha,n}^T = \mathsf{rot}_{n-\alpha,n}$ and $\mathsf{rot}_{\alpha,n}^k = \mathsf{rot}_{k\alpha \bmod n,n}$.

Theorem 4.1. *For any $1 \le \alpha < n$ with $\gcd(\alpha, n) = 1$, and any 2-monotone distribution D over $\{0,1\}^n$ with min-entropy $k > 1$, it holds that*

$$H_2(D^{(n)}_{\mathsf{rot}_{\alpha,n}}) \ge n(1 - \log_2(1 + 2^{-2k+2})) \approx n(1 - 2^{-2k+2}) ,$$

$$H_{\min}(D^{(n)}_{\mathsf{rot}_{\alpha,n}}) \ge n(1 - \log_2(1 + 2^{-k+1})) \approx n(1 - 2^{-k+1}) .$$

Theorem 4.1 provides some basic theoretical justification for the use of $\mathsf{rot}_{\alpha,n}$ as a condenser. It shows rotation provably condenses to $\Omega(n)$ bits entropy within n steps.

Note that the theorem works for *any* rotation with $\gcd(\alpha, n) = 1$, which means it also works for rotation whose rotation number is $\alpha = 1$. Although we typically think of rotation by 1 as the worst condenser (and we will show this in the next section), our result shows it can still condense 2-monotone distributions to linear entropy within n steps. What's more, the proof of Theorem 4.1 immediately generalizes to *any* cyclic permutation,[9] so that any cyclic permutation can condense to $\Omega(n)$ bits of entropy within n steps. (In particular, this can be applied to the cyclic permutation tor_n that we define in Section 6.)

Proof. Let $A := \mathsf{rot}_{\alpha,n}$. For $\boldsymbol{w} \in \{0,1\}^n$, we say that $\boldsymbol{w}$ hits $\boldsymbol{e}_0$ if $w_0 = 1$, and say that $(\boldsymbol{w}, A^T\boldsymbol{w}, \ldots, (A^T)^{n-1}\boldsymbol{w})$ hits $\boldsymbol{e}_0$ j times if there are j choices of i such that $(A^T)^i\boldsymbol{w}$ hits $\boldsymbol{e}_0$. By Lemma 3.2, for every $\boldsymbol{w}$ hitting $\boldsymbol{e}_0$, it holds that $|\widehat{D}(\boldsymbol{w})| \le 2^{-k+1}$.

Claim 4.2. *For $\boldsymbol{w} \in \{0,1\}^n$, $(\boldsymbol{w}, A^T\boldsymbol{w}, \ldots, (A^T)^{n-1}\boldsymbol{w})$ hits $\boldsymbol{e}_0$ exactly $|\boldsymbol{w}|$ times.*

Proof. It suffices to notice that for every i, there exists a distinct $0 \le j < n$ such that $(A^T)^j\boldsymbol{e}_i = \boldsymbol{e}_0$ since $A = \mathsf{rot}_{\alpha,n}$ and $\gcd(\alpha, n) = 1$. For such an i and j, $(A^T)^j\boldsymbol{w}$ hits $\boldsymbol{e}_0$ if and only if $w_i = 1$. Therefore, the total number of hits is exactly the number of non-zero coordinates in $\boldsymbol{w}$. ☐

Given the claim, we note that

$$|\widehat{D^{(n)}_A}(\boldsymbol{w})| = \left|\prod_{i=0}^{n-1} \widehat{D}((A^T)^i\boldsymbol{w})\right| \le \prod_{i:(A^T)^i\boldsymbol{w} \text{ hits } \boldsymbol{e}_0} |\widehat{D}((A^T)^i\boldsymbol{w})| \le (2^{-k+1})^{|\boldsymbol{w}|}$$

where the equality is by Claim 2.4, and the last inequality is by the claim and Lemma 3.2. Therefore,

$$\sum_{\boldsymbol{w}\in\{0,1\}^n} |\widehat{D^{(n)}_A}(\boldsymbol{w})|^2 \le \sum_{\boldsymbol{w}\in\{0,1\}^n} (2^{-k+1})^{2|\boldsymbol{w}|} = \sum_{j=0}^{n} \binom{n}{j} (2^{-k+1})^{2j} = (1 + 2^{-2k+2})^n ,$$

$$\sum_{\boldsymbol{w}\in\{0,1\}^n} |\widehat{D^{(n)}_A}(\boldsymbol{w})| \le \sum_{\boldsymbol{w}\in\{0,1\}^n} (2^{-k+1})^{|\boldsymbol{w}|} = \sum_{j=0}^{n} \binom{n}{j} (2^{-k+1})^{j} = (1 + 2^{-k+1})^n .$$

Finally, by Corollary 2.3, we have

$$H_2(D^{(n)}_A) = n - \log(\sum_{\boldsymbol{w}\in\{0,1\}^n} |\widehat{D^{(n)}_A}(\boldsymbol{w})|^2) \ge n - \log(1 + 2^{-2k+2})^n$$

[9] A cyclic permutation is a permutation $\sigma : [n] \to [n]$ such that for all $x \in [n]$, $\sigma^i(x) = x$ if and only if n divides i.

$$H_{\min}(D_A^{(n)}) \geq n - \log\Big(\sum_{\boldsymbol{w}\in\{0,1\}^n} |\widehat{D_A^{(n)}}(\boldsymbol{w})|\Big) \geq n - \log(1+2^{-k+1})^n$$

which imply the desired conclusion. □

5 Comparing different rotations and permutations

In this section, we show how to compare the performance of different permutations on two-monotone distributions. For a permutation $\pi : [n] \to [n]$, we write A_π for the linear transformation over $\{0,1\}^n$ defined by

$$A_\pi(x_1, \dots, x_n) := (x_{\pi(1)}, \dots, x_{\pi(n)}) \ .$$

I.e., A_π permutes coordinates of a vector $\boldsymbol{x}$ by π. We slightly abuse notation and write $D_\pi^{(\ell)}$ to denote $D_{A_\pi}^{(\ell)}$.

We show that the rate of convergence of the condenser associated with a permutation π when run on two-monotone distributions with min-entropy k is governed by what we call the *covering number* $C_{\pi,k}$.

To get some intuition behind the covering number, consider the simple case when we want to extract from the distribution D that is uniform on $[2^k]$ (or, in terms of bit strings, uniform on $\{0,1\}^k \times \{0\}^{n-k}$). Notice that $D_\pi^{(m)}$ is the distribution that is uniform over the space spanned by $\{\boldsymbol{e}_i \ : \ \exists 0 \leq j < k, 0 \leq \ell < m, \ \pi^\ell(j) = i\}$.

In particular, the distribution $D_\pi^{(m)}$ has full entropy n if and only if $\{\pi^\ell(j) \ : \ 0 \leq j < k, 0 \leq \ell < m\}$ is the full set of coordinates $[n]$. We call the minimal such m the covering number of π, and the above discussion shows that it arises naturally in this context.

Definition 5.1 (Covering number). *For a permutation $\pi : [n] \to [n]$, and an integer $1 \leq k \leq n$, the covering number $C_{\pi,k}$ is the smallest natural number m such that*

$$\{\pi^\ell(j) \ : \ 0 \leq j < k, 0 \leq \ell < m\} = [n]$$

where $\pi^i = \pi \circ \pi^{i-1}$ for $i \geq 1$, π^0 is the identity function. (If no such m exists, we say $C_{\pi,k} = \infty$.)

To get some intuition for the covering number, first notice that we must have $C_{\pi,k} \geq \lceil n/k \rceil$. (This corresponds to the fact that we need at least $\lceil n/k \rceil$ sources with k bits of entropy each in order to have any hope of extracting n bits of entropy.) Of course, the covering number can be much worse than this, e.g., $C_{\mathsf{rot}_{1,n},k} = n - k + 1$, which is the worst possible.

Also, notice that $C_{\mathsf{rot}_{k,n},k} = \lceil n/k \rceil$. In other words, the optimal covering number $\lceil n/k \rceil$ is always achieved for fixed k by rotation by exactly k bits. (This suggests that, if your input is "nice enough," e.g., 2-monotone, and you happen to *know* it has k bits of entropy, then rotating by $\alpha = k$ is a good idea. And, indeed, this is the case.) So, every choice of α has an optimal covering number for at least one choice of k, and we will not be able to unambiguously say that one choice of α is the "best". Still, the performance of different choices of α over all $1 \leq k \leq n$ does vary considerably.

As we explained above, the covering number arises naturally when considering how quickly we can extract from the uniform distribution on $[2^k]$, which is perhaps the simplest 2-monotone distribution.

The following theorem shows that the covering number actually characterizes how quickly we converge to a high-entropy distribution for *any* 2-monotone distribution. In the proof, "we lose a factor of two in k," so that we need to take at least $C_{\pi,k/2}$ steps, rather than $C_{\pi,k}$ steps. And, we are of course only able to condense, not to extract. But, otherwise the result is tight. In Section 7.2, we show empirically that roughly $C_{\pi,k}$ steps is already enough for very strong condensing for natural distributions, suggesting that the factor of two loss is an artifact of the proof. Put together, the empirical data and the theoretical justification below strongly suggest that

The covering number $C_{\pi,k}$ is the right measure of how well a permutation π condenses for natural real-world distributions.

Theorem 5.2. *Let D be a two-monotone distribution with min-entropy at least k for some $k \geq 2$. Let $\pi : [n] \to [n]$ be a permutation with covering number $m := C_{\pi,k'}$, where $k' := \lfloor k/2 \rfloor$. Then for any $\ell \geq m$,*

$$H_2(D_\pi^{(\ell)}) \geq n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k' - k\lfloor \ell/m \rfloor}) \approx n(1 - 2^{k/2 - k\ell/m}) \ ,$$

$$H_{\min}(D_\pi^{(\ell)}) \geq n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k' - (k/2)\lfloor \ell/m \rfloor}) \approx n(1 - 2^{k/2 - k\ell/(2m)}) \ .$$

Furthermore, there exists a two-monotone distribution D with min-entropy k such that for all $1 \leq \ell < C_{\pi,k}$

$$H_{\min}(D_\pi^{(\ell)}) = H_2(D_\pi^{(\ell)}) \leq n - (C_{\pi,k} - \ell) \ .$$

In Appendix A, we prove the following better lower bound on condensing (using the techniques developed in Section 7), showing that the results in Theorem 5.2 are quite tight. In fact, the distribution that we use to prove the theorem is exactly the same as the distribution that we use in our empirical results in Section 7.2. (This distribution arises naturally in this context.)

Theorem 5.3. *For every integer $1 \leq s \leq n$, there is a 2-monotone distribution D (in fact, a monotone distribution) with min-entropy $k > s - 1 + 1/2^s$ such that*

$$H_2(D_\pi^{(\ell)}) \leq n \cdot (1 - 2^{-s^2\ell/n - 4\ell/n}) \approx n(1 - 2^{-k^2\ell/n - 4\ell/n})$$
$$H_{\min}(D_\pi^{(\ell)}) \leq n \cdot (1 - 2^{-s^2\ell/(2n) - 2\ell/n}) \approx n(1 - 2^{-k^2\ell/(2n) - 2\ell/n}) \ .$$

Notice how close these bounds are to the bounds in Theorem 5.2 with $m := n/k$.

5.1 Proof of Theorem 5.2

Proof. The "furthermore" part of the theorem is trivial. Indeed, it holds for the uniform distribution over $[2^k]$, which is clearly 2-monotone with min-entropy k. So, it remains only to prove the first statement.

Let $k' = \lfloor k/2 \rfloor$ and let $B_0, B_1, \ldots, B_{m-1}$ be a partition of $[n]$ such that $B_i \subseteq \pi^i([k'])$. Observe that $0 \leq |B_i| \leq k'$. We use $\boldsymbol{w}_S$ to denote the projection of $\boldsymbol{w}$ onto $S \subseteq [n]$. Let b be either 1 or 2.

Claim 5.4. $\sum_{\boldsymbol{w}} |\widehat{D_\pi^{(\ell)}}(\boldsymbol{w})|^b \leq \sum_{\boldsymbol{w}} |\widehat{D_\pi^{(m)}}(\boldsymbol{w})|^{b\lfloor \ell/m \rfloor}$.

Proof.

$$\begin{aligned}
\sum_{\boldsymbol{w}} |\widehat{D_\pi^{(\ell)}}(\boldsymbol{w})|^b &= \sum_{\boldsymbol{w}} \prod_{i=0}^{\ell-1} |\widehat{D}((A_\pi^T)^i \boldsymbol{w})|^b \\
&\le \sum_{\boldsymbol{w}} \prod_{j=0}^{\lfloor \ell/m \rfloor -1} \prod_{i=0}^{m-1} |\widehat{D}((A_\pi^T)^{jm+i} \boldsymbol{w})|^b \\
&= \sum_{\boldsymbol{w}} \prod_{j=0}^{\lfloor \ell/m \rfloor -1} |\widehat{D_\pi^{(m)}}((A_\pi^T)^{jm} \boldsymbol{w})|^b \\
&\le \prod_{j=0}^{\lfloor \ell/m \rfloor -1} \Big(\sum_{\boldsymbol{w}} |\widehat{D_\pi^{(m)}}((A_\pi^T)^{jm} \boldsymbol{w})|^{b\lfloor \ell/m \rfloor} \Big)^{1/\lfloor \ell/m \rfloor} \\
&= \prod_{j=0}^{\lfloor \ell/m \rfloor -1} \Big(\sum_{\boldsymbol{w}'} |\widehat{D_\pi^{(m)}}(\boldsymbol{w}')|^{b\lfloor \ell/m \rfloor} \Big)^{1/\lfloor \ell/m \rfloor} \\
&= \sum_{\boldsymbol{w}} |\widehat{D_\pi^{(m)}}(\boldsymbol{w})|^{b\lfloor \ell/m \rfloor}
\end{aligned}$$

where the first line is by Claim 2.4, the fourth line is by Claim B.1 and the fifth line is because $(A_\pi^T)^{jm}$ is a permutation over $\{0,1\}^n$. □

Claim 5.5. $|\widehat{D_\pi^{(m)}}(\boldsymbol{w})|^{b\lfloor \ell/m \rfloor} \le 2^{-a_{\boldsymbol{w}} \cdot (bk/2)\lfloor \ell/m \rfloor}$*, where* $a_{\boldsymbol{w}} := |\{i \;:\; \boldsymbol{w}_{B_i} \neq \mathbf{0}\}|$.

Proof. We say that $\boldsymbol{w}$ hits $[k']$ if there exists an $i \in [k']$ such that $w_i = 1$. By Lemma 3.2, for any 2-monotone distribution with min-entropy at least k, and any $\boldsymbol{w}$ which hits $[k']$, it holds that

$$|\widehat{D}(\boldsymbol{w})| \le 2^{(k'-1)+1-k} \le 2^{-k/2} .$$

For $\boldsymbol{w}$, let $S_{\boldsymbol{w}} := \{i \in [m] : (A_\pi^T)^i \boldsymbol{w} \text{ hits } [k']\}$. Observe that, if $\boldsymbol{w}_{B_i} \neq \mathbf{0}$, because $B_i \subseteq \pi^i([k'])$, then

$$\big((A_\pi^T)^i \boldsymbol{w}\big)_{[k']} = \boldsymbol{w}_{\pi^i([k'])} \neq \mathbf{0}$$

where $\pi^i([k']) := \{\pi^i(j) : j \in [k']\}$. I.e., $(A_\pi^T)^i \boldsymbol{w}$ hits $[k']$. Therefore, $|S_{\boldsymbol{w}}| \ge a_{\boldsymbol{w}}$. Moreover,

$$\begin{aligned}
|\widehat{D_\pi^{(m)}}(\boldsymbol{w})|^{b\lfloor \ell/m \rfloor} &= \prod_{i=0}^{m-1} |\widehat{D}((A_\pi^T)^i(\boldsymbol{w}))|^{b\lfloor \ell/m \rfloor} \\
&\le \prod_{i \in S_{\boldsymbol{w}}} |\widehat{D}((A_\pi^T)^i \boldsymbol{w})|^{b\lfloor \ell/m \rfloor} \\
&\le 2^{-|S_{\boldsymbol{w}}| \cdot (bk/2)\lfloor \ell/m \rfloor} \\
&\le 2^{-a_{\boldsymbol{w}} \cdot (bk/2)\lfloor \ell/m \rfloor}
\end{aligned}$$

as needed. □

We then prove Theorem 5.2 given above claims.

$$\begin{aligned}
\sum_{\boldsymbol{w}} |\widehat{D_\pi^{(\ell)}}(\boldsymbol{w})|^b &\le \sum_{\boldsymbol{w}} |\widehat{D_\pi^{(m)}}(\boldsymbol{w})|^{b\lfloor \ell/m \rfloor} \\
&\le \sum_{\boldsymbol{w}} 2^{-a_{\boldsymbol{w}}\cdot(bk/2)\lfloor \ell/m \rfloor} \\
&= \sum_{S\subseteq[m]} \binom{m}{|S|} \sum_{\boldsymbol{w}:\{i:\boldsymbol{w}_{B_i}\neq 0\}=S} 2^{-|S|\cdot(bk/2)\lfloor \ell/m \rfloor} \\
&= \sum_{S\subseteq[m]} \binom{m}{|S|} 2^{-|S|\cdot(bk/2)\lfloor \ell/m \rfloor} \prod_{i\in S}(2^{|B_i|}-1) \\
&= \sum_{S\subseteq[m]} \binom{m}{|S|} \prod_{i\in S}\left((2^{|B_i|}-1)2^{-(bk/2)\lfloor \ell/m \rfloor}\right) \\
&= \prod_{i=0}^{m-1}\left(1+(2^{|B_i|}-1)\cdot 2^{-(bk/2)\cdot\lfloor \ell/m \rfloor}\right)
\end{aligned}$$

where the first inequality is by Claim 5.4, and the second inequality is by Claim 5.5. Furthermore, by Claim B.2, when $0 \le |B_i| \le k'$ and $\sum_{i=0}^{m-1} |B_i| = n$,

$$\prod_{i=0}^{m-1}\left(1+(2^{|B_i|}-1)\cdot 2^{-(bk/2)\cdot\lfloor \ell/m \rfloor}\right) \le \left(1+(2^{k'}-1)\cdot 2^{-(bk/2)\cdot\lfloor \ell/m \rfloor}\right)^{\lfloor n/k' \rfloor+1}.$$

Finally, by Corollary 2.3, we have

$$H_2(D_\pi^{(\ell)}) \ge n - \log\left(\sum_{\boldsymbol{w}} |\widehat{D_\pi^{(\ell)}}(\boldsymbol{w})|^2\right) \ge n - (\lfloor \frac{n}{k'} \rfloor + 1)\cdot \log\left(1+2^{k'-k\lfloor \ell/m \rfloor}\right),$$

$$H_{\min}(D_\pi^{(\ell)}) \ge n - \log\left(\sum_{\boldsymbol{w}} |\widehat{D_\pi^{(\ell)}}(\boldsymbol{w})|\right) \ge n - (\lfloor \frac{n}{k'} \rfloor + 1)\cdot \log\left(1+2^{k'-(k/2)\lfloor \ell/m \rfloor}\right).$$

as needed. □

5.2 Covering numbers of different rotations when $n = 32, 64$

In Figures 3 and 4, we show the covering numbers of rotations with different rotation number α when $n = 32, 64$. In both cases, we compare the rotation numbers α chosen by Microsoft ($\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$) with other rotations that also perform well. (In Section 6, we show a different cyclic permutation, which is not a rotation, but has optimal covering number $C_{n,k} = n/k$ when both n and k are powers of two.) We also compare the covering numbers with the natural lower bound of $\lceil n/k \rceil$.

6 A new recommendation: bit-reversed rotation

Our characterization in terms of the covering number suggests the following "greedy" construction of a permutation with small covering number. Recall that we can write a cyclic permutation π in

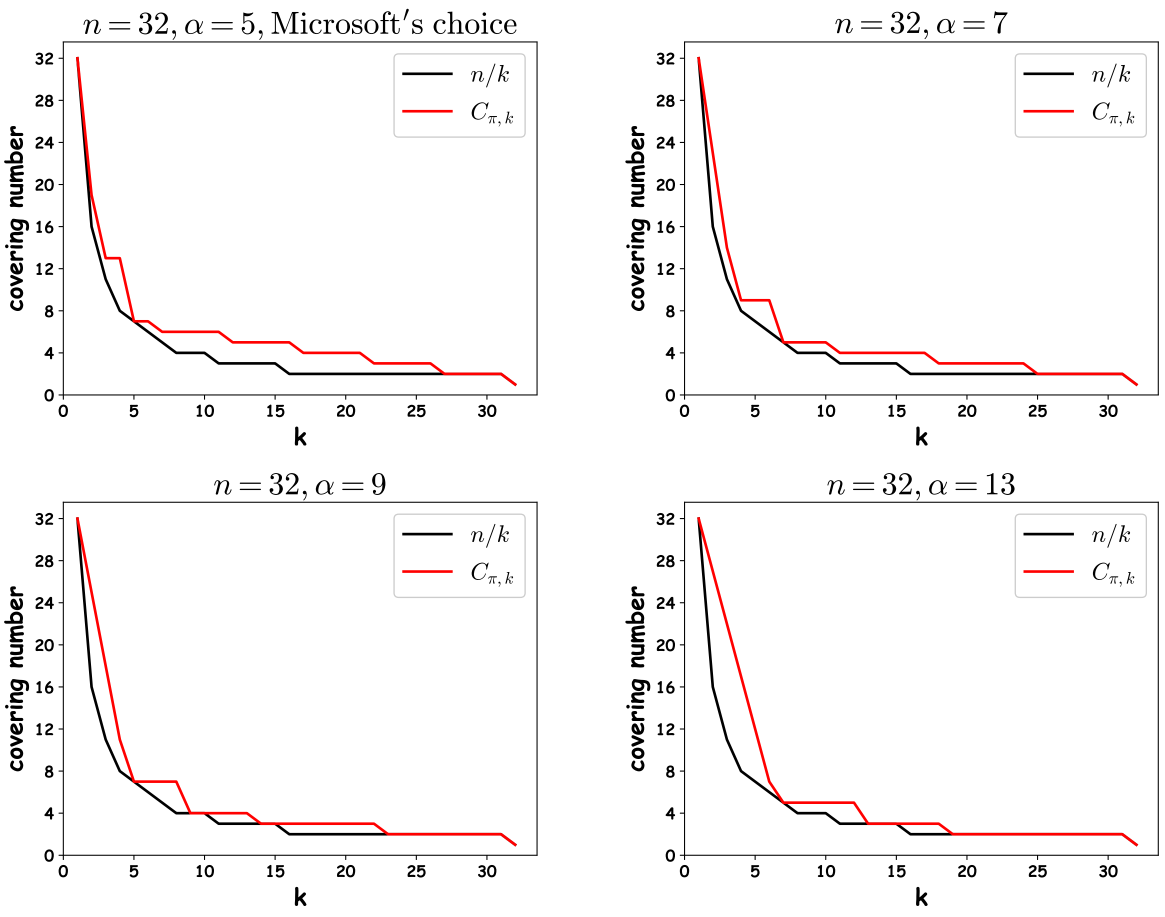

Figure 3: Covering numbers of different rotations when $n = 32$

cycle notation as

$$a_0 := 0 \to a_1 := \pi(0) \to a_2 := \pi(\pi(0)) \to \cdots \to a_{n-1} := \pi^{n-1}(0) \to a_n := 0\ .$$

So, suppose that n is a power of two, and suppose that we want to build some permutation $\pi : [n] \to [n]$ such that $C_{\pi,2} = n/2$ is as small as possible. Notice that this holds if and only if $a_{n/2} = 1$. I.e., "0 and 1 should be maximally far apart on the cycle":

$$a_0 = 0 \to a_1 \to \cdots \to a_{n/2-1} \to a_{n/2} = 1 \to a_{n/2+1} \to \cdots \to a_{n-1} \to a_n = 0\ .$$

Similarly, $C_{\pi,4} = n/4$ if and only if $\{a_{n/4}, a_{n/2}, a_{3n/4}\} = \{1, 2, 3\}$, i.e., if and only if "0, 1, 2, and 3 are maximally far apart on the cycle" so that no two of them are within distance less than $n/4$. Therefore if we simultaneously have $C_{\pi,2} = n/2$ *and* $C_{\pi,4} = n/4$, then the permutation must have either the form

$$a_0 = 0 \to \cdots \to a_{n/4} = 2 \to \cdots \to a_{n/2} = 1 \to \cdots \to a_{3n/4} = 3 \to \cdots \to 0, \qquad (1)$$

or

$$a_0 = 0 \to \cdots \to a_{n/4} = 3 \to \cdots \to a_{n/2} = 1 \to \cdots \to a_{3n/4} = 2 \to \cdots \to 0. \qquad (2)$$

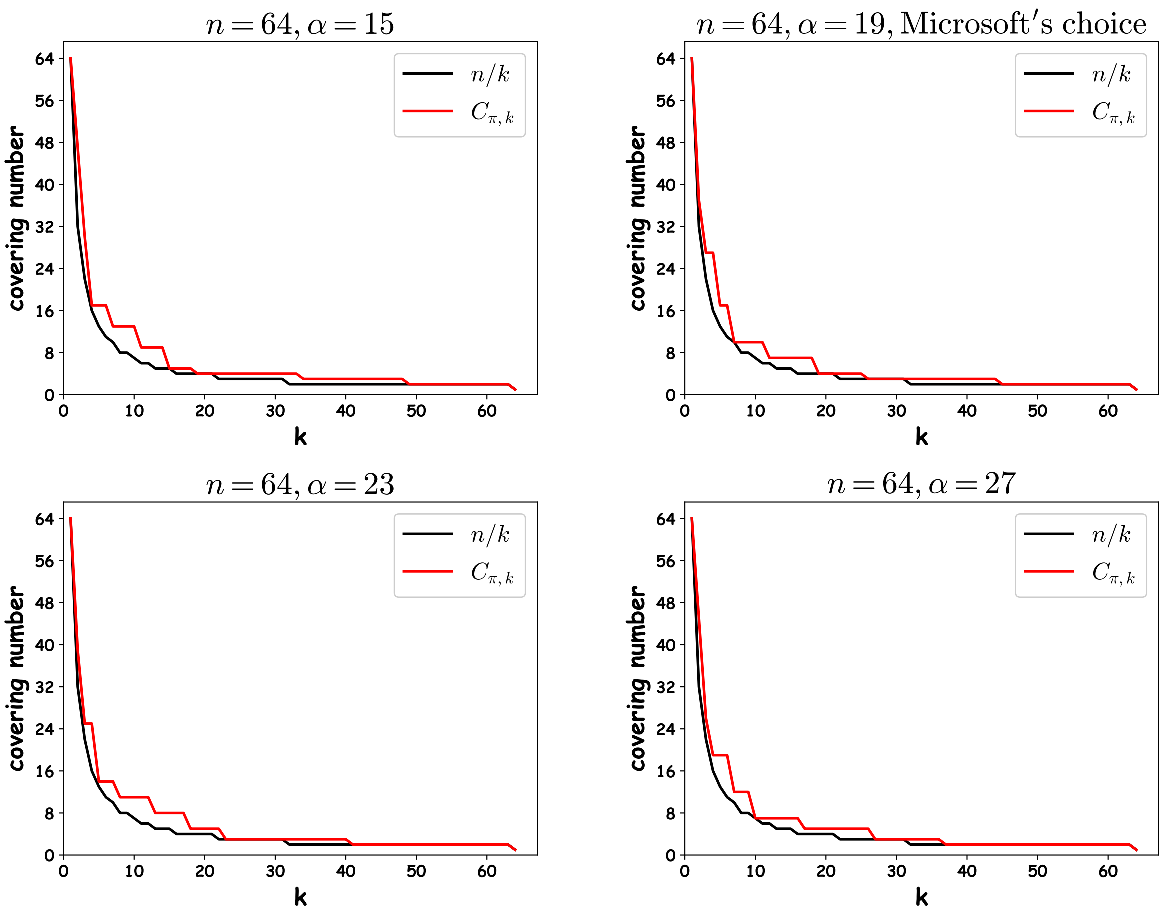

Figure 4: Covering number of different rotations when $n = 64$

Continuing in this way, we see that we can build a cyclic permutation $\pi : [n] \to [n]$ such that $C_{\pi,2^a} = n/2^a$ for all integers $0 \le a \le \log_2 n$. In fact, we get a family of permutations (where the different members of the family vary as in Eqs. (1) and (2)), which represents all permutations that satisfy $C_{\pi,2^a} = n/2^a$ for all a. And, it is not hard to see that every such permutation has covering numbers given by $C_{\pi,k} = n/k'$, where $k' := 2^{\lfloor \log_2 k \rfloor}$ is the largest power of two smaller than k. (We prove this carefully below for one particular member of the family.)

Since all such permutations are essentially identical from our perspective, we choose one with a particularly elegant description. This elegant description might also help with efficient implementations. In particular, our choice, which we call bit-reversed rotation, is obtained by conjugating $\mathsf{rot}_{1,n}$ with the well-studied and very efficient bit-reversal permutation.

Definition 6.1 (Bit-reversed rotation). *For a power of two $n = 2^a$, the bit-reversal permutation $\sigma_n : [n] \to [n]$ is defined by*

$$\sigma_n\Big(b_0 + 2b_1 + \cdots + 2^{a-1}b_{a-1}\Big) = b_{a-1} + 2b_{a-2} + \cdots + 2^{a-1}b_0$$

for $b_i \in \{0, 1\}$. (E.g., $\sigma_8(3) = 6$, and $\sigma_{16}(10) = 5$.) Notice that σ_n is an involution, i.e., $\sigma_n^{-1} = \sigma_n$.

We nevertheless sometimes write σ_n^{-1} when this seems more natural.

Then, the bit-reversed rotation $\mathsf{tor}_n : [n] \to [n]$ *(*tor *is* rot *"reversed") is given by* $\mathsf{tor}_n := \sigma_n^{-1} \circ \mathsf{rot}_{1,n} \circ \sigma_n$*. E.g., in cyclic notation,* tor_8 *is*

$$0 \to 4 \to 2 \to 6 \to 1 \to 5 \to 3 \to 7 \to 0 \; .$$

Equivalently, tor_n *can be defined recursively via the recurrence* $\mathsf{tor}_{2n}(i+n) = \mathsf{tor}_n(i)$ *for* $i < n$ *together with the identity* $\mathsf{tor}_{2n}(i) = i + n$*. (These two rules as simply describe binary addition, except with the bits reversed. I.e., "if the highest-order bit is 0, set it to 1; if it is 1, then set it to 0 and perform the same operation on the remaining bits.")*

Theorem 6.2. *For a power of two n and $1 \le k \le n$,*

$$C_{\mathsf{tor}_n,k} = n/k' \le 2n/k \; ,$$

where $k' := 2^{\lfloor \log_2 k \rfloor}$ is the largest power of two that is no larger than k.

Proof. The result follows from the recurrence relation $C_{\mathsf{tor}_{2n},k} = 2C_{\mathsf{tor}_n,k}$ for $k \le n$, together with two base cases, $C_{\mathsf{tor}_n,n} = 1$ and $C_{\mathsf{tor}_n,\ell} = 2$ for $n/2 \le \ell < n$.

The first base case, $C_{\mathsf{tor}_n,n} = 1$ is trivial. The second base case $C_{\mathsf{tor}_n,\ell} = 2$ for $n/2 \le \ell < n$ follows the simple observation that for all $i < n/2$, $\mathsf{tor}_n(i) \ge n/2$. This in particular implies that $C_{\mathsf{tor}_n,n/2} = 2$, and since $1 < C_{\mathsf{tor}_n,\ell} \le C_{\mathsf{tor}_n,n/2}$ for $n/2 \le \ell < n$, we must have $C_{\mathsf{tor}_n,\ell} = 2$ for all such ℓ.

To see the recurrence relation, notice that the recursive formula for tor_n implies that $\mathsf{tor}_{2n}^2(i) = \mathsf{tor}_n(i)$ for $i < n$. Applying this identity repeatedly gives $\mathsf{tor}_{2n}^{2\ell}(i) = \mathsf{tor}_n^{\ell}(i)$. Similarly, $\mathsf{tor}_{2n}^{2\ell+1}(i) = \mathsf{tor}_n^{\ell}(i)+n$. It follows that $i \in \{\mathsf{tor}_n^{\ell}(0), \ldots, \mathsf{tor}_n^{\ell}(k-1)\}$ if and only if $i, i+n \in \{\mathsf{tor}_{2n}^{2\ell}(0), \ldots, \mathsf{tor}_{2n}^{2\ell}(k-1), \mathsf{tor}_{2n}^{2\ell+1}(0), \ldots, \mathsf{tor}_{2n}^{2\ell+1}(k-1)\}$, which immediately implies the recurrence relation. □

Applying Theorem 5.2 immediately yields the following corollary, which shows that the bit-reversed rotation yields quite a good condenser. (In Section 7.2, we show empirical results that suggest even better performance, suggesting that the factor of 2 loss in k' is unnecessary.)

Corollary 6.3. *For any power of two n, and any 2-monotone distribution D with min-entropy at least $k \ge 2$, then for any $\ell \ge m$*

$$H_2(D^{(\ell)}_{\mathsf{tor}_n}) \ge n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k'-k\lfloor \ell/m \rfloor}) \approx n \cdot (1 - 2^{-k^2\ell/(2n)}) \; ,$$

$$H_{\min}(D^{(\ell)}_{\mathsf{tor}_n}) \ge n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k'-(k/2)\lfloor \ell/m \rfloor}) \approx n \cdot (1 - 2^{-k^2\ell/(4n)}) \; ,$$

where $k' := \lfloor k/2 \rfloor$, and $m := n/2^{\lfloor \log_2 k' \rfloor}$ is the smallest power of two that is no smaller than n/k'.

In Figure 5, we plot the covering numbers $C_{\mathsf{tor}_n,k}$ together with $C_{\mathsf{rot}_{\alpha,n},k}$ for comparison.

6.1 A brief note on efficient implementation

We leave it to practitioners to determine whether bit-reversed rotation can be implemented efficiently enough for their applications. However, we do note two things. First, we note that the bit-reversal permutation σ_n on which bit-reversed rotation is based is well-studied (in part because of its relationship with algorithms for the Fast Fourier Transform), with many fast implementations known (see, e.g., [Kar96]).

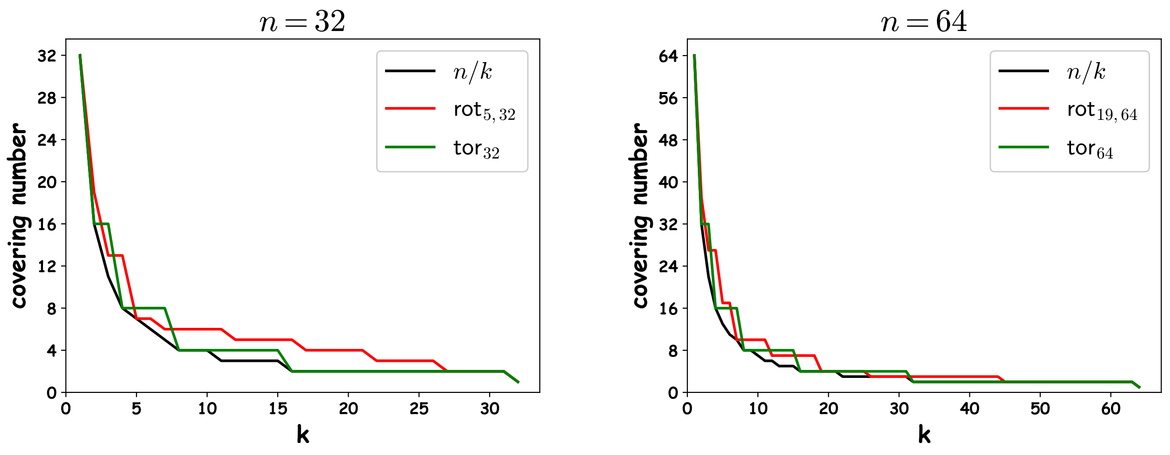

Figure 5: The covering numbers of bit-reversed rotation and the two rotations $\mathsf{rot}_{5,32}$ and $\mathsf{rot}_{19,64}$ used by Microsoft. The n/k line represents the best possible value of $\lceil n/k \rceil$.

Second, recall that we have defined our extractor according to the state update procedure $S_{i+1} \leftarrow \mathsf{tor}_n(S_i) \oplus X = \sigma_n(\mathsf{rot}_{1,n}(\sigma_n(S_i))) \oplus X$, where σ_n is the bit-reversal permutation. (Here, we have used the fact that $\sigma_n = \sigma_n^{-1}$, to replace $\sigma_n^{-1} \circ \mathsf{rot}_{1,n} \circ \sigma_n$ with simply $\sigma_n \circ \mathsf{rot}_{1,n} \circ \sigma_n$, since for implementation it seems quite useful to observe that these are the same map.) However, we note that one can equivalently use the rule $S'_{i+1} \leftarrow \mathsf{rot}_{1,n}(S'_i) \oplus \sigma_n(X)$. I.e., one can simply perform the bit-reversal permutation on the input X and rotate the state S' by one. It is then easy to see that this rule maintains the invariant $S_i = \sigma_n(S'_i)$, and in particular, that S_i and S'_i have the same entropy. Therefore, one can use the second rule instead of the first, which could improve efficiency by replacing two applications of σ_n with a single application.[10]

7 Examples of natural distributions and some computational results

Here, we list some natural distributions, all of which are 2-monotone. We then compute exactly the number of steps necessary to condense for the special case of the exponential distribution, which has a particularly nice form that makes such exact computation feasible.

Discrete Gaussian. For $s > 0$, we write $D_{\mathbb{Z},s}$ for the discrete Gaussian distribution over the integers with parameter s, i.e., defined by

$$\Pr_{X \sim D_{\mathbb{Z},s}}[X = z] = \frac{\exp(-\pi z^2/s^2)}{\sum_{z'=-\infty}^{\infty} \exp(-\pi (z')^2/s^2)}$$

for all integers $z \in \mathbb{Z}$.

Similarly, for $\sigma > 0$, the exponential distribution $E_{\mathbb{Z},\sigma}$ with parameter σ is the distribution over

[10]More generally, for any invertible linear transformations A, B, one can replace the rule $S_{i+1} \leftarrow B^{-1}AB(S_i) \oplus X$ with the equivalent rule $S'_{i+1} \leftarrow AS'_i \oplus BX$. This maintains the invariant $S_i = B^{-1}S'_i$.

$\mathbb{Z}_{\geq 0}$ defined by

$$\Pr_{X \sim D_\sigma}[X = z] = \frac{\exp(-z/\sigma)}{\sum_{z'=0}^{\infty} \exp(-z'/\sigma)}$$

for all integers $z \geq 0$.

Uniform distribution over an interval. For $0 \leq N_1 < N_2 \leq 2^n$, let U_{N_1,N_2} be the distribution over $\{0,1\}^n$ obtained by sampling an integer $N_1 \leq X < N_2$ uniformly at random (interpreted as a bit string as above).

Shifted exponential distribution. For any $\sigma \geq 1$ and any integer $0 \leq N < 2^n$, let $E_{\sigma,N}$ be the distribution over $\{0,1\}^n$ obtained by sampling an integer Y from an exponential distribution with parameter σ and setting $X := N + Y \bmod 2^n$ (and interpreting this as a bit string).

Shifted discrete Gaussian distribution. For any $s \geq 1$ and any integer $0 \leq N < 2^n$, let $D_{s,N}$ be the distribution over $\{0,1\}^n$ obtained by sampling an integer Y from the discrete Gaussian distribution $D_{\mathbb{Z},s}$ with parameter σ and setting $X := N + Y \bmod 2^n$ (and interpreting this as a bit string).

Claim 7.1. *U_{N_1,N_2}, $E_{\sigma,N}$, and $D_{s,N}$ are all 2-monotone.*

Proof. This fact is immediate for the uniform distribution U_{N_1,N_2}, by taking the monotone intervals $\{N_1, \ldots, N_2 - 1\}$ and $\{N_2 - 1, \ldots, \ldots, 2^n + N_1 - 1\}$. Similarly, for $E_{\sigma,N}$, we can take the intervals $\{N-1, N\}$ and $\{N, \ldots, 2^n + N - 1\}$. For the Gaussian $D_{s,N}$, this follows from the fact that the function $t \mapsto \sum_{z \in \mathbb{Z}} e^{-\pi(z-t)^2}$ has maxima at $t \in \mathbb{Z}$, minima at $t \in \mathbb{Z} + 1/2$ and no other critical points, which one can verify, e.g., using the Poisson summation formula. □

7.1 Entropy of product distributions under permutations

Below, we show an *exact* formula for the min-/collision entropy resulting from applying our permutation-based condensers to a product distribution. This exact formula is of course very useful, as it allows us to easily compute the min-/collision entropy of the state of our extractor, without directly computing the sum of 2^n Fourier coefficients. Indeed, in Section 7.2, we use this formula to show empirically that our extractor performs similarly as well with the unshifted exponential distribution—a product distribution.

Theorem 7.2. *Let D be a product distribution over $\{0,1\}^n$ with $\Pr_{\boldsymbol{x} \sim D}[X_i = 0] = (1 + \varepsilon_i)/2$ for $\varepsilon_i \geq 0$. Then, for any cyclic permutation $\pi : [n] \to [n]$,*

$$H_2(D_\pi^{(\ell)}) = n - \sum_{i=0}^{n-1} \log_2 \left(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}^2\right),$$

$$H_{\min}(D_\pi^{(\ell)}) = n - \sum_{i=0}^{n-1} \log_2 \left(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}\right).$$

Proof. We have

$$\widehat{D_\pi^{(\ell)}}(\boldsymbol{w}) = \prod_{j=0}^{\ell-1} \widehat{D}(\pi^j(\boldsymbol{w})) = \prod_{j=0}^{\ell-1} \prod_{w_i=1} \widehat{D}(\boldsymbol{e}_{\pi^j(i)}) = \prod_{j=0}^{\ell-1} \prod_{w_i=1} \varepsilon_{\pi^j(i)} \geq 0$$

where the second equality is because D is a product distribution. Therefore,

$$D_\pi^{(\ell)}(\boldsymbol{x}) = \frac{1}{2^n} \sum_{\boldsymbol{w} \in \{0,1\}^n} \widehat{D_\pi^{(\ell)}}(\boldsymbol{w})(-1)^{\langle \boldsymbol{x}, \boldsymbol{w} \rangle} \le \frac{1}{2^n} \sum_{\boldsymbol{w} \in \{0,1\}^n} \widehat{D_\pi^{(\ell)}}(\boldsymbol{w}) = D_\pi^{(\ell)}(0) \ .$$

Moreover,

$$\sum_{\boldsymbol{w}} |\widehat{D_\pi^{(\ell)}}(\boldsymbol{w})|^2 = \prod_{i=0}^{n-1} \Big(1 + \prod_{j=0}^{\ell-1} |\widehat{D}(\boldsymbol{e}_{\pi^j(i)})|^2\Big) = \prod_{i=0}^{n-1} \Big(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}^2\Big) \ ,$$

$$\sum_{\boldsymbol{w}} \widehat{D_\pi^{(\ell)}}(\boldsymbol{w}) = \prod_{i=0}^{n-1} \Big(1 + \prod_{j=0}^{\ell-1} \widehat{D}(\boldsymbol{e}_{\pi^j(i)})\Big) = \prod_{i=0}^{n-1} \Big(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}\Big) \ .$$

The equality for $H_2(D_\pi^{(\ell)})$ follows from Corollary 2.3, and the equality for $H_{\min}(D_\pi^{(\ell)})$ follows from

$$H_{\min}(D_\pi^{(\ell)}) = \log_2(1/D_\pi^{(\ell)}(0)) = \log_2(2^n / \sum_{\boldsymbol{w} \in \{0,1\}^n} \widehat{D_\pi^{(\ell)}}(\boldsymbol{w})) \ .$$

□

Corollary 7.3. *For any $\sigma \ge 1$, let $D := E_\sigma$ be the distribution over $\{0,1\}^n$ obtained by sampling an integer Y from an exponential distribution with parameter σ and setting $X := Y \bmod 2^n$ (and interpreting this as a bit string), as described above. Then, for any cyclic permutation $\pi : [n] \to [n]$,*

$$H_2(D_\pi^{(\ell)}) = n - \sum_{i=0}^{n-1} \log_2 \Big(1 + \Big(\prod_{j=0}^{\ell-1} \frac{1 - \exp(-2^{\pi^j(i)}/\sigma)}{1 + \exp(-2^{\pi^j(i)}/\sigma)}\Big)^2\Big)$$

$$H_{\min}(D_\pi^{(\ell)}) = n - \sum_{i=0}^{n-1} \log_2 \Big(1 + \prod_{j=0}^{\ell-1} \frac{1 - \exp(-2^{\pi^j(i)}/\sigma)}{1 + \exp(-2^{\pi^j(i)}/\sigma)}\Big) \ .$$

Proof. For any $0 \le x < 2^n$,

$$\Pr_{X \sim D}[X = x] = \frac{\sum_{z \ge 0} \exp(-(x + 2^n z)/\sigma)}{\sum_{z \ge 0} \exp(-z/\sigma)} = C_\sigma \cdot \prod_i \exp(-2^i x_i/\sigma) \ ,$$

where

$$C_\sigma := \frac{\sum_{z \ge 0} \exp(-2^n z/\sigma)}{\sum_{z \ge 0} \exp(-z/\sigma)} \ .$$

I.e., D is a product distribution. From the above expression, we see that

$$\Pr_{X \sim D}[X_i = 0] = \exp(2^i/\sigma) \cdot \Pr_{X \sim D}[X_i = 1] \ .$$

The desired conclusion then follows from Theorem 7.2 with

$$\varepsilon_i = \frac{1 - \exp(-2^i/\sigma)}{1 + \exp(-2^i/\sigma)} \ .$$

□

7.2 Computational results for $n = 32$ and $n = 64$

Finally, we use the formula from Corollary 7.3 to directly compute the number of samples needed to condense to nearly full entropy from the exponential distribution with different starting entropy and different permutations. At a high level, these results confirm that the covering number $C_{\pi,k}$ provides a good estimate for the number of steps needed to condense.

In more detail, in Figure 6, we show the exact number of samples needed to condense to $n-1$ bits of *collision* entropy from the exponential distribution with k bits of entropy for $\mathsf{rot}_{5,32}$, $\mathsf{rot}_{19,64}$ (i.e., both rotations used by Microsoft), tor_{32}, and tor_{64}. We plot these relative to the associated covering numbers, showing the close relationship between the covering number and the number of samples needed. Indeed, Figure 6 is quite striking in terms of how well the covering number $C_{\pi,k}$ matches the number accumulation steps needed.

Figure 7 shows the same numbers for min-entropy. While the result is less striking in this case, it is still clear from the figure that the covering number $C_{\pi,k}$ more-or-less governs the accumulation behavior.

Figure 8 presents the same information as Figure 6 (i.e., number of steps needed to accumulate $n-1$ bits of collision entropy) but shows a direct comparison between $\mathsf{rot}_{5,32}$ and tor_{32}, and between $\mathsf{rot}_{19,64}$ and tor_{64}. Figure 9 does the same with min-entropy. In particular, the two figures show that bit-reversal rotation compares quite favorably with the permutations used by Microsoft in practice, suggesting that it might be a good alternative.

Figure 10 presents the same information as the previous plots but shows a direct comparison between the number of steps needed to accumulate $n-1$ bits of collision entropy as opposed to the number of steps needed to accumulate $n-1$ bits of min-entropy. This shows that collision entropy might be a more useful metric than min-entropy—particularly in the case when the number of input bits k is small.

Finally, Figure 11 shows the accumulation of entropy over time—rather than just the number of steps needed to converge to $n-1$ bits specifically—from an exponential distribution with 2 bits of min-entropy. This can be useful, e.g., for applications in which significantly fewer bits of entropy (e.g., $n/2$ bits) are needed. It is evident from the figure that H_2 outperforms $H_{\min}$ consistently—not just for the specific threshold of $n-1$ bits. Figures 12 and 13 show the same data for a variety of different values for the starting entropy k. We note that the difference between H_2 an $H_{\min}$ is most pronounced in the low-entropy regime (as is also reflected by our theoretical results).

8 Bigger picture

In this work we abstracted out and analysed the fast entropy accumulation procedures found in modern RNGs. We can now combine our results with prior RNG literature [DPR+13, DSSW14, GT16, Hut16, CDKT19] to get a better big picture guarantee of the resulting RNG, but we start with some observations.

First, every RNG so far used a different (and often incompatible) modeling of the security of the slow refresh procedure. So, it's not clear what is the "right" model for slow refresh—let alone a hybrid fast/slow model. The good news is that all of the slow refresh models share one thing in common—their rate of convergence depends only on either the overall collision/min-entropy that they received. So, the fact that our work guarantees entropic output from a fast refresh seems like a good start in trying to unify the two models.

Indeed, our results *do* (trivially) combine with every prior RNG work, and for concreteness we state one such combination below (focusing on the slow refresh RNG work of [DSSW14], but other combinations are done analogously). The main issue with such naive combinations is that they appear to only work with a relatively weak RNG adversary, which must output independent (but not necessarily identical) samples from the family of two-monotone distributions $\mathcal{D}_{k,n}$, for (fixed but unknown) min-entropy k per sample. We show a concrete example of a combined slow-refresh and fast-refresh procedure in Appendix D, but we stress that this is meant only as a proof of concept and that we do not claim that the model used in Appendix D is the "right" model.

As a positive, this is already a highly non-trivial and quite interesting model. For example, very related "constant entropy rate" adversaries were mentioned by Fergusson and Schneier [FS03]—and later formalized by [DSSW14]—in their design and analysis of Fortuna, which directly led to the Windows 10 RNG [Fer19]. On the negative side, prior work on slow refresh [DPR$^+$13, DSSW14, GT16, Hut16]
[CDKT19] worked hard to give a lot of power to the RNG attacker, including the ability to output correlated samples, and drastically change the entropy of each sample (subject only to providing enough entropy overall). Thus, it is unfortunate that the naive composition that follows from using our work did not use all these powerful security guarantees of the slow-refresh procedures, and resulted in a much weaker overall attacker.

We note that, while the naive composition currently does not capture the ability of the distribution sampler to change its entropy parameter k, it is clear that the final RNG is at least somewhat resilient and easily adaptable to this change, as the RNG design does not use the knowledge of k, and simultaneously provides good guarantees for all k. So it feels the actual hybrid slow-/fast-refresh RNG is much more robust that the current composition states. If nothing else, the resulting RNGs are basically what is used in the real world, and these RNGs appear to work well against practical entropy sources. In particular, while it is great that the standards for the slow refresh procedure in the literature are very high, the existing entropy sources (e.g., modeling timing of interrupts) appear to be much less adversarial, and likely lie somewhere in between the independent 2-monotone sources modeled in this paper and the very general classes handled in the literature on slow refresh procedures.

In summary, we view our current work as only the starting point in trying to understand and model the overall security of the composed RNG, and believe that finding the "right" model to combine fast refresh with slow refresh is a great avenue for future work.

Acknowledgments. Yevgeniy Dodis — Partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546.

Siyao Guo — Supported by Shanghai Eastern Young Scholar Program SMEC-0920000169. Parts of this work were done while visiting the Centre for Quantum Technologies, National University of Singapore.

Noah Stephens-Davidowitz — Some of this work was done at MIT supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS1718161, Microsoft Faculty Fellowship and an MIT/IBM grant. Some of this work was done at the Simons Institute in Berkeley.

References

[BH05] Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In *CCS*, 2005.

[BIW04] B. Barak, R. Impagliazzo, and A. Wigderson. Extracting randomness using few independent sources. In *FOCS*, 2004.

[BTRS02] Ziv Bar-Yossef, Luca Trevisan, Omer Reingold, and Ronen Shaltiel. Streaming computation of combinatorial objects. In *CCC*, 2002.

[CDKT19] Sandro Coretti, Yevgeniy Dodis, Harish Karthikeyan, and Stefano Tessaro. Seedless fruit is the sweetest: Random number generation, revisited. In *CRYPTO*, 2019.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

[CZ19] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. *Annals of Mathematics*, 189(3):653–705, 2019.

[DPR$^+$13] 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 *CCS*, 2013.

[DSSW14] 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 *CRYPTO*, 2014.

[Fer13] Niels Ferguson. Private communication, 2013.

[Fer19] Niels Ferguson. The windows 10 random number generation infrastructure. https://www.microsoft.com/security/blog/2019/11/25/going-in-depth-on-the-windows-10-random-number-generation-infrastructure/, 2019. [Online; posted October 2019].

[FS03] Niels Ferguson and Bruce Schneier. *Practical Cryptography.* John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2003.

[GT16] Peter Gazi and Stefano Tessaro. Provably robust sponge-based prngs and kdfs. In *Eurocrypt*, 2016.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hut16] Daniel Hutchinson. A robust and sponge-like PRNG with improved efficiency. In *SAC*, 2016.

[Kar96] Alan H. Karp. Bit reversal on uniprocessors. *SIAM Rev*, 38:1–26, 1996.

[KRVZ11] 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.

[KSF99] John Kelsey, Bruce Schneier, and Niels Ferguson. Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In *SAC*, 1999.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[RR99] Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *STOC*, 1999.

[RSW00] O. Reingold, R. Shaltiel, and A. Wigderson. Extracting randomness via repeated condensing. In *FOCS*, 2000.

[sup] Supplementary material: covering numbers for all rotations with $n = 32$ and $n = 64$. http://noahsd.com/rotation_extractor_supplement.zip.

[Wik04] Wikipedia. /dev/random. http://en.wikipedia.org/wiki//dev/random, 2004. [Online; accessed 09-February-2014].

A Proof of Theorem 5.3

We now prove Theorem 5.3. In fact, our example distribution is simply the exponential distribution $D := E_\sigma$, as defined in Section 7, where $\sigma := 2^s$. (This distribution arises naturally in this context as a monotone distribution that nearly minimizes the ratio $\varepsilon_i/\varepsilon_{i-1}$ of consecutive biases.) The result then follows by applying appropriate bounds to the closed-form expressions in Corollary 7.3 for $H_{\min}(D_\pi^{(\ell)})$ and $H_2(D_\pi^{(\ell)})$.

In particular, from the corollary, we see that

$$H_{\min}(D) = n - \sum_{i=0}^{n-1} \log_2(1+\varepsilon_i) \geq s - \sum_{i=0}^{s-1} \log_2(1+\varepsilon_i) \ ,$$

where

$$\varepsilon_i := \frac{1-\exp(-2^i/\sigma)}{1+\exp(2^i/\sigma)} \ .$$

Then, for $i \leq s-1$,

$$\log_2(1+\varepsilon_i) < 1-\exp(-2^i/\sigma) < 2^i/\sigma \ .$$

Therefore,

$$H_{\min}(D) > s - \sum_{i=0}^{s-1} 2^i/\sigma = s - 1 + 1/\sigma \ ,$$

as claimed.

We now turn to the upper bound on $H_2(D_\pi^{(\ell)})$. Again by Corollary 7.3, we have

$$H_2(D_\pi^{(\ell)}) = n - \sum_{i=0}^{n-1} \log_2\Big(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}^2\Big) \leq n - n \cdot \prod_{i=0}^{n-1} \log_2\Big(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}^2\Big)^{1/n} \ ,$$

where we have used the inequality $\sum a_i/n \geq \prod a_i^{1/n}$, valid for $a_1, \ldots, a_n \geq 0$. Next, recall that for all $0 \leq x \leq 1$, $\log_2(1+x) \geq x$. Therefore,

$$\prod_{i=0}^{n-1} \log_2\Big(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}^2\Big)^{1/n} \geq \prod_{i=0}^{n-1}\prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}^{2/n} = \prod_{i=0}^{n-1} \varepsilon_i^{2\ell/n} \ .$$

Now, notice that

$$\prod_{i=0}^{n-1} \varepsilon_i \geq \frac{1}{2} \cdot \prod_{i=0}^{s} \varepsilon_i \ .$$

Finally, notice that for $i \leq s$, $\varepsilon_i \geq 2^{i-2}/\sigma$, so that

$$\prod_{i=0}^{s} \varepsilon_i \geq \sigma^{-s} \cdot \prod_{i=0}^{s} 2^{i-2} = (1/2)\cdot(4/\sigma)^s \cdot 2^{s(s+1)/2} \geq \sigma^{-s/2}/2 \ .$$

The upper bound for H_2 follows.

The proof of the upper bound for $H_{\min}$ is essentially identical, with ε_i^2 replaced by ε_i.

B Additional missing proofs

Claim B.1. *For any integers $n, m \geq 1$, it holds that*

$$\sum_{i\in[n]} \prod_{j\in[m]} |a_{i,j}| \leq \prod_{j\in[m]} \Big(\sum_{i\in[n]} |a_{i,j}|^m \Big)^{1/m}$$

where $a_{i,j}$ are arbitrary real numbers.

Proof. We prove the claim by induction on m. When $m = 1$, it holds trivially for any $n \geq 1$. Suppose the statement holds for some $m \geq 1$ (and any $n \geq 1$). Then, we consider $m + 1$. By Hölder's inequality, for any $p, q \geq 1$ such that $1/p + 1/q = 1$, it holds that

$$\begin{aligned}\sum_{i\in[n]} \prod_{j\in[m+1]} |a_{i,j}| &= \sum_{i\in[n]} |a_{i,m}| \prod_{j\in[m]} |a_{i,j}| \\ &\leq \Big(\sum_{i\in[n]} \Big(\prod_{j\in[m]} |a_{i,j}| \Big)^p \Big)^{1/p} \cdot \Big(\sum_{i\in[n]} |a_{i,m}|^q \Big)^{1/q} .\end{aligned}$$

By the induction hypothesis and choosing $p = 1 + 1/m$ and $q = m + 1 = (1 - 1/p)^{-1}$, we have

$$\begin{aligned}\sum_{i\in[n]} \prod_{j\in[m+1]} |a_{i,j}| &\leq \Big(\sum_{i\in[n]} \Big(\prod_{j\in[m]} |a_{i,j}| \Big)^p \Big)^{1/p} \cdot \Big(\sum_{i\in[n]} |a_{i,m}|^q \Big)^{1/q} \\ &= \Big(\sum_{i\in[n]} \prod_{j\in[m]} |a_{i,j}|^p \Big)^{1/p} \cdot \Big(\sum_{i\in[n]} |a_{i,m}|^q \Big)^{1/q} \\ &\leq \prod_{j\in[m]} \Big(\sum_{i\in[n]} |a_{i,j}|^{pm} \Big)^{1/(pm)} \cdot \Big(\sum_{i\in[n]} |a_{i,m}|^q \Big)^{1/q} \\ &= \prod_{j\in[m+1]} \Big(\sum_{i\in[n]} |a_{i,j}|^{m+1} \Big)^{1/(m+1)} ,\end{aligned}$$

as needed. ☐

Claim B.2. *For $0 \leq c \leq 1$, and $0 \leq a_0, \ldots, a_{m-1} \leq k'$ such that $\sum_{i\in[m]} a_i = n$,*

$$\prod_{i=0}^{m-1} \big(1 + (2^{a_i} - 1) \cdot c\big) \leq \big(1 + (2^{k'} - 1) \cdot c\big)^{\lfloor n/k' \rfloor + 1} .$$

Proof. Suppose that $a_0, \ldots, a_{m-1}$ maximizes

$$f(a_0, \ldots, a_{m-1}) = \prod_{i=0}^{m-1} \big(1 + (2^{a_i} - 1) \cdot c\big) .$$

Without loss of generality, we assume $a_0 \geq a_1 \geq \cdots \geq a_{m-1}$. Let $t = \lfloor n/k' \rfloor$. It suffices to show that $a_i = k'$ for $i \in [t]$, $a_t = n - k't$, and $a_i = 0$ for $i > t$ is an optimal solution. Then,

$$f(a_0, \ldots, a_{m-1}) \leq \big(1 + (2^{k'} - 1) \cdot c\big)^{t+1} .$$

Suppose that there exists $i \in [t]$ such that $a_i < k'$. Then $\sum_{i\in[t]} a_i < n$, and there exists $j \geq t$ such that $a_j > 0$. Observe that for any $b \geq 0$, $0 \leq c \leq 1$ and $a_i \geq a_j$, it holds that

$$\begin{aligned}
&(1 + (2^{a_i+b} - 1) \cdot c)(1 + (2^{a_j-b} - 1) \cdot c) - (1 + (2^{a_i} - 1) \cdot c)(1 + (2^{a_j} - 1) \cdot c) \\
&= (2^{a_i} - 2^{a_j-b})(2^b - 1)(c - c^2) \\
&\geq 0.
\end{aligned}$$

So we can increase a_i by $b := \min\{k' - a_i, a_j\}$ and decrease a_j by b such that $a_0, \ldots, a_{m-1}$ is feasible and $f(a_0, \ldots, a_{m-1})$ doesn't decrease. By repeating this process, we will reach the solution that $a_i = k'$ for $i \in [t]$, $a_t = n - k't$, and $a_i = 0$ for $i > t$, which is therefore an optimal solution as well. □

C From collision entropy and min-entropy to statistical distance

The statistical distance between two distributions D_1 and D_2 over $\{0,1\}^n$ is

$$\mathsf{SD}(D_1, D_2) := \frac{1}{2} \cdot \sum_{\boldsymbol{x}\in\{0,1\}^n} |D_1(\boldsymbol{x}) - D_2(\boldsymbol{x})| \, .$$

We say D_1 is ε-close to D_2 if $\mathsf{SD}(D_1, D_2) \leq \varepsilon$. We say D has *smooth min-entropy* k, denoted as $H^{\varepsilon}_{\min}(D) := k$, if D is ε-close to some D' with $H_{\min}(D') = k$.

The collision entropy of a distribution D over $\{0,1\}^n$ provide useful bounds for its (smooth) min-entropy and statistical distance from uniform distribution U over $\{0,1\}^n$.

Fact C.1. *$H_{\min}(D) \leq H_2(D) \leq 2H_{\min}(D)$, and $H^{\varepsilon}_{\min}(D) \geq H_2(D) - \log_2(1/\varepsilon)$.*

Fact C.2. *$\mathsf{SD}(D, U) \leq \frac{1}{2} \cdot \sqrt{2^{n-H_2(D)} - 1}$, and $\mathsf{SD}(D, U) \leq 2^{n-H_{\min}(D)} - 1$.*

D A concrete example combining fast and slow refresh

For illustrative purposes, we will use the already-mentioned "constant-rate" RNG work of [DSSW14] (described in Section 6, Theorem 5 of that work) in order to show one possible way of combining a slow-refresh model with our fast-refresh model. This work gives the simplest model to state, and also attempts to model the Windows 10 RNG [Fer19]. The RNG analyzed by [DSSW14] solves a very difficult "premature next" problem, by splitting its slow refresh state S into multiple "entropy pools" $S = (S_1, \ldots, S_p)$, and cleverly selecting which pool S_i to use with each given each new fast-refresh output R_j. For our purposes, we abstract out these details, only focusing on the final security guarantee from [DSSW14].

Existing Slow Refresh Result. Let λ be the security parameter, which roughly models the amount of min-entropy needed to result in a random state by cryptographically hashing some distribution Y having min-entropy λ. Let $0 < \gamma \leq \lambda$ be the (unknown) entropy of a constant-rate (slow-refresh) attacker $\mathcal{A}$, and q be the upper bound on the number of such slow-refresh calls that $\mathcal{A}$ can make. Then the RNG from [DSSW14] recovers from state compromise after at most $C_{\lambda,q}(\gamma)$ slow-refresh calls made by $\mathcal{A}$,[11] where:

[11]Even if $\mathcal{A}$ can call the "premature next" oracle defined by [DSSW14], whose details are unimportant here.

$$C_{\lambda,q}(\gamma) \le 2.1 \log_2 q \cdot (\lambda/\gamma)$$

In other words, while at least (λ/γ) slow-refresh calls are needed just to give λ bits of fresh entropy, the Fortuna (slow-refresh) RNG is multiplicative factor $2.1 \log_2 q$ away from this minimal amount.

Combined Slow+Fast Refresh Result. To add fast-refresh to the mix, imagine we have c registers $R^1, \ldots, R^c$ (each having n bits), and now our new (also constant-rate) attacker $\mathcal{B}$ can choose arbitrary independent samples $X_1, X_2, \ldots$ from the family of 2-monotone distributions $\mathcal{D}_{k,n}$, where k is min-entropy of each such sample. Assume the registers are updated by the fast-refresh rule $R \leftarrow A_\pi R \oplus X$ in a round-robin manner, using an appropriate permutation π. Moreover, all c registers $R = (R^1, \ldots, R^c)$ are emptied into the slow-refresh procedure after each was used ℓ times.[12] This means after $c\ell$ calls to the fast-refresh procedure by $\mathcal{B}$, we can use Theorem 5.2 to conclude that each register R^i has min-entropy

$$\gamma^*(k) = H_{\min}(R^i) \ge n - \lceil n/k' \rceil \cdot \log_2(1 + 2^{k' - k\lfloor \ell/m \rfloor/2}) \approx n(1 - 2^{-k\ell/2m}) ,$$

where $k' := \lfloor k/2 \rfloor$ and $m := C_{\pi,k'}$ is the covering number of π w.r.t. k'.

This means that our fast-refresh attacker $\mathcal{B}$ translates the original slow-refresh attacker $\mathcal{A}$ with entropy parameter $\gamma = c\gamma^*(k)$, and the number of induced slow-fresh queries $q = Q/(c\ell)$, where Q is the total number of fast-refresh queries (i.e., actual "interrupts") output by $\mathcal{B}$.

Putting it all together, and remembering that each slow refresh happens after $c\ell$ fast-refreshes (i.e., interrupts), we get the following composition. For given k, λ, Q, ℓ, c, and n, the composed RNG recovers from compromise after $D_{\lambda,Q,\ell,c,n}(k)$ fast-refresh calls, where:

$$D_{\lambda,Q,\ell,c,n}(k) \le 2.1 \log_2 \left(\frac{Q}{c\ell}\right) \cdot \left(\frac{\lambda\ell}{\gamma^*(k)}\right) \approx 2.1 \log_2 \left(\frac{Q}{c\ell}\right) \cdot \left(\frac{\lambda\ell}{n(1 - 2^{-k\ell/2m})}\right)$$

where $k' := \lfloor k/2 \rfloor$ and $m := C_{\pi,k'}$ is the covering number of π w.r.t. k'.

A different way to read this result is to remember that, at the minimum, we need λ bits of entropy, and each sample gives us k bits. So the overhead of this provable result against the information-theoretic minimum λ/k is roughly

$$2.1 \log_2 \left(\frac{Q}{c\ell}\right) \cdot \left(\frac{k\ell}{n(1 - 2^{-k\ell/2m})}\right)$$

The first term $2.1 \log_2 \left(\frac{Q}{c\ell}\right)$ is an artifact from the slow-refresh work of [DSSW14], so the extra overhead coming from this work is an additional multiplicative factor

$$F_{n,\ell}(k) \approx \frac{k\ell}{n(1 - 2^{-k\ell/2m})}$$

Namely, after ℓ calls per register, we could have hoped to get $k\ell$ entropy, but due to the superfast nature of our accumulator (and the fact that it has bounded size n), we "only" got roughly $n(1 - 2^{-k\ell/2m})$ bits of entropy. Moreover, this is just the *provable upper bound* we derive in this work. As we show in Section 7.2, for many setting of parameters the number is actually much closer to the information-theoretic optimum $k\ell$, meaning the fast-refresh analysis adds almost no overhead in practice: the "real" $F_{n,\ell}(k) \approx 1$.

[12]This assumption is made for simplicity of the bound, and can be generalized easily.

E Additional figures

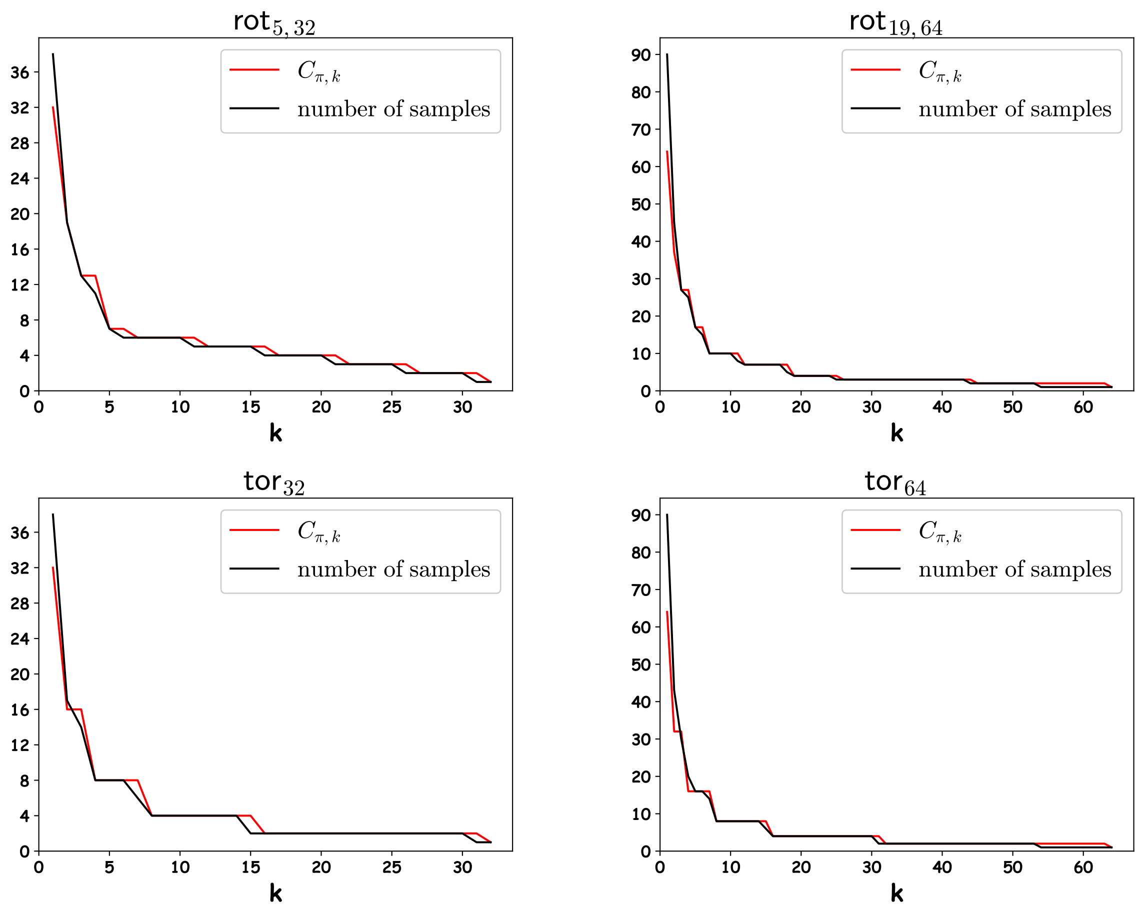

Figure 6: The number of steps needed for different permutations to accumulate $n-1$ bits of collision entropy from an exponential distribution with k bits of min-entropy plotted relative to the covering number $C_{\pi,k}$. (The fact that the red line is often not visible shows that the number of samples is often exactly equal to the covering number. The two rotations chosen $\mathsf{rot}_{5,32}$ and $\mathsf{rot}_{19,64}$ are the ones used by Microsoft. Figure 6 shows the analogous numbers for min-entropy. Figure 8 shows the same data plotted together for more easy comparison.)

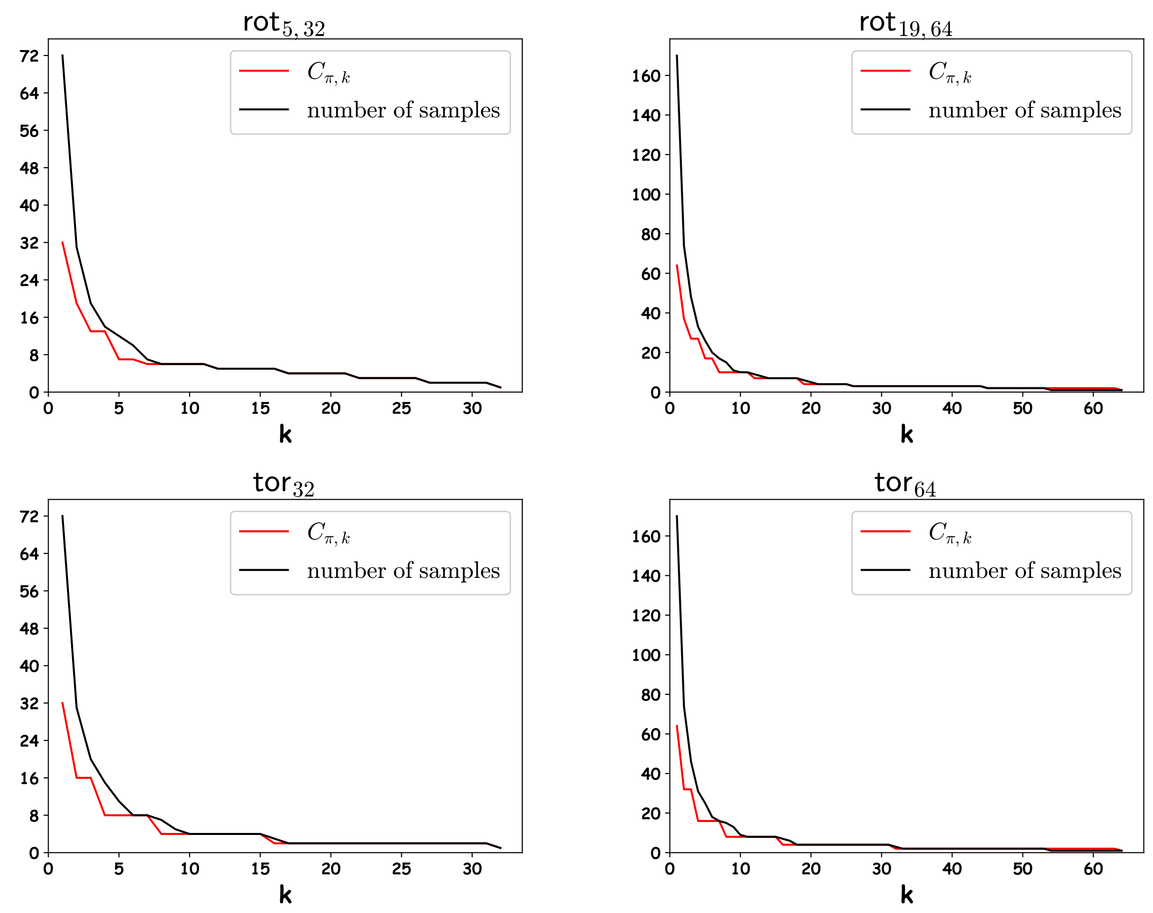

Figure 7: The number of steps needed for different permutations to accumulate $n-1$ bits of min-entropy from an exponential distribution with k bits of min-entropy plotted relative to the covering number $C_{\pi,k}$. (The fact that the red line is often not visible shows that the number of samples is often exactly equal to the covering number. The two rotations chosen $\mathsf{rot}_{5,32}$ and $\mathsf{rot}_{19,64}$ are the ones used by Microsoft. Figure 6 shows the analogous numbers for collision entropy. Figure 9 shows the same data plotted together for more easy comparison.)

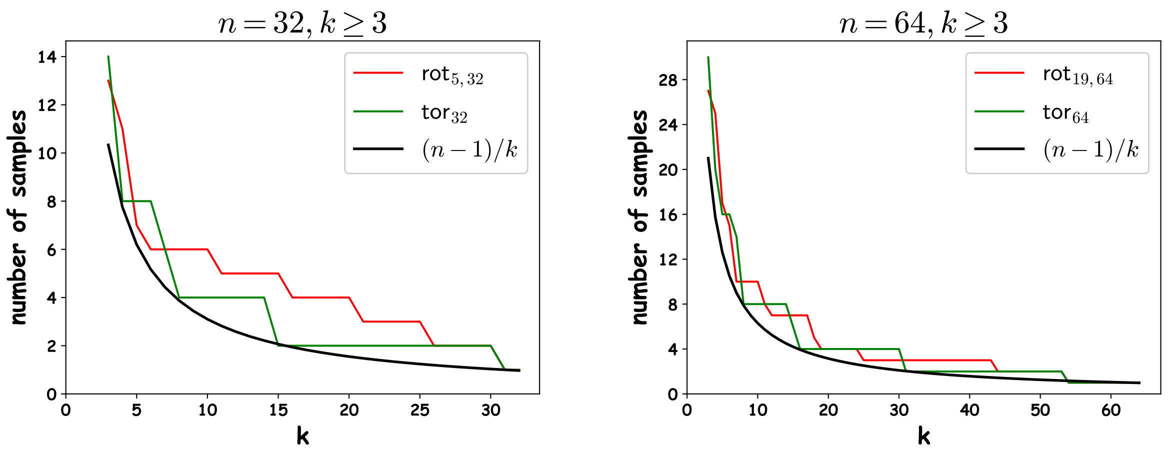

Figure 8: The number of steps needed for different permutations to accumulate $n-1$ bits of collision entropy from an exponential distribution with k bits of min-entropy. The $(n-1)/k$ line represents an approximately theoretical information-theoretically optimal condenser. (The two rotations chosen $\mathsf{rot}_{5,32}$ and $\mathsf{rot}_{19,64}$ are the ones used by Microsoft. Figure 9 shows the analogous numbers for min-entropy. Figure 6 shows the same data plotted against the relevant covering numbers $C_{\pi,k}$.)

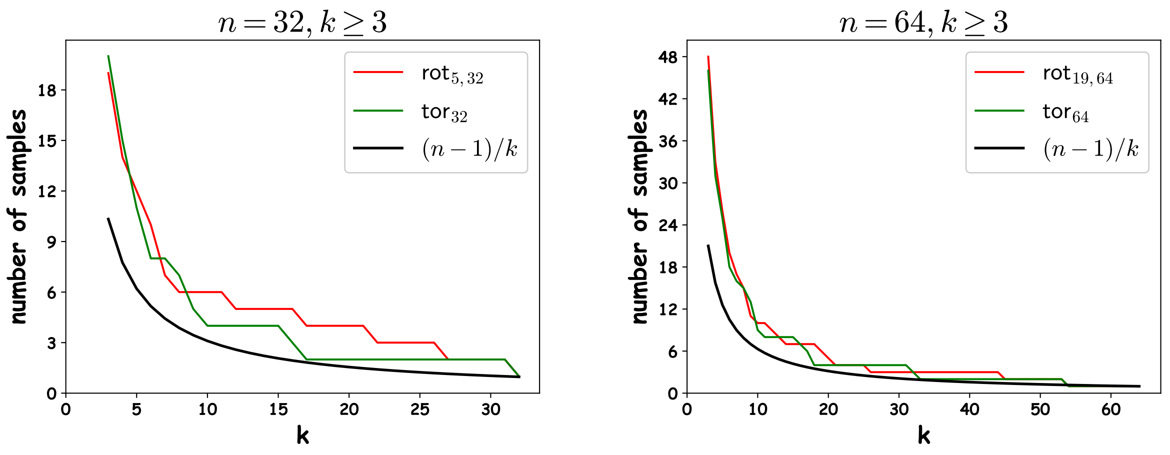

Figure 9: The number of steps needed for different permutations to accumulate $n-1$ bits of min-entropy from an exponential distribution with k bits of min-entropy. The $(n-1)/k$ line represents a theoretical information-theoretically optimal condenser. (The two rotations chosen $\mathsf{rot}_{5,32}$ and $\mathsf{rot}_{19,64}$ are the ones used by Microsoft. Figure 8 shows the analogous numbers for min-entropy. Figure 7 shows the same data plotted against the relevant covering numbers $C_{\pi,k}$.)

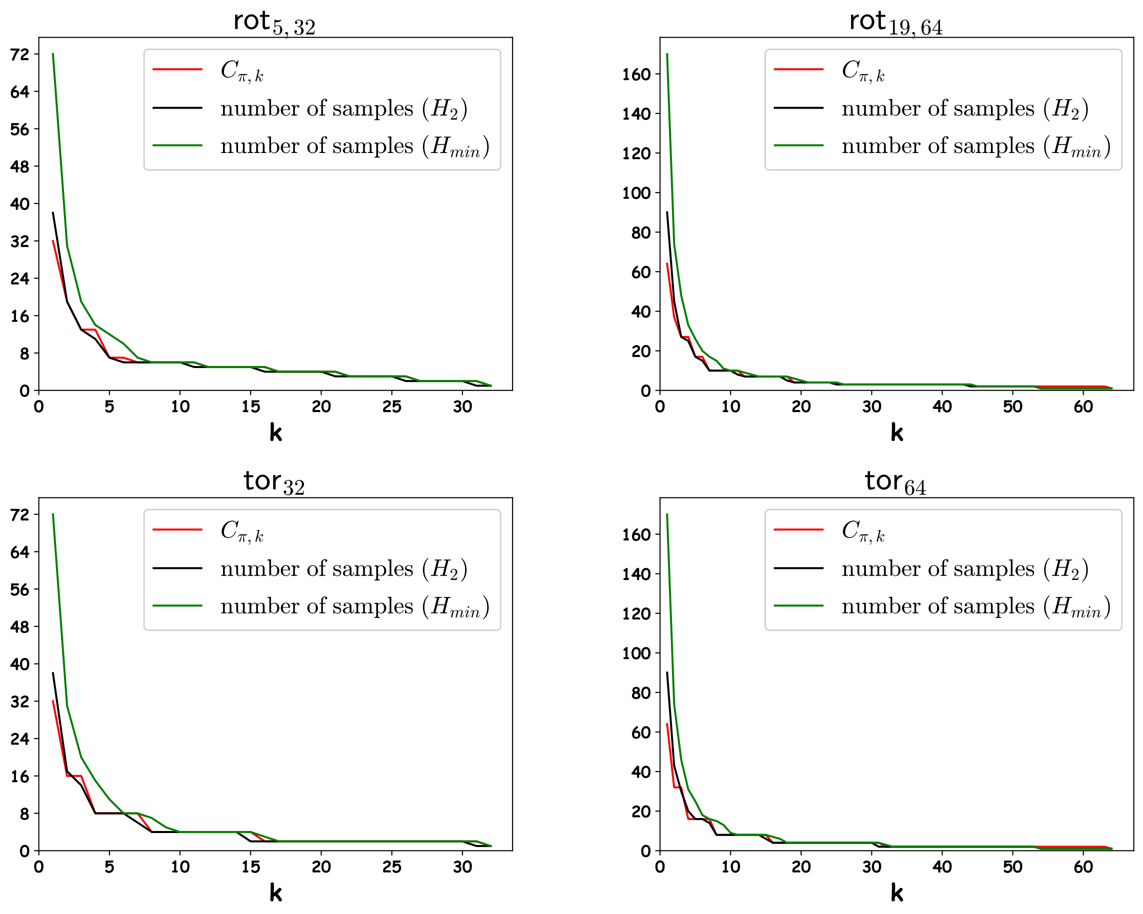

Figure 10: The number of steps needed for different permutations to accumulate $n-1$ bits of min-entropy ($H_{\min}$) from an exponential distribution with k bits of min-entropy, and the number of steps needed to accumulate $n-1$ bits of *collision* entropy (H_2) the same distribution. (The two rotations chosen $\mathsf{rot}_{5,32}$ and $\mathsf{rot}_{19,64}$ are the ones used by Microsoft.)

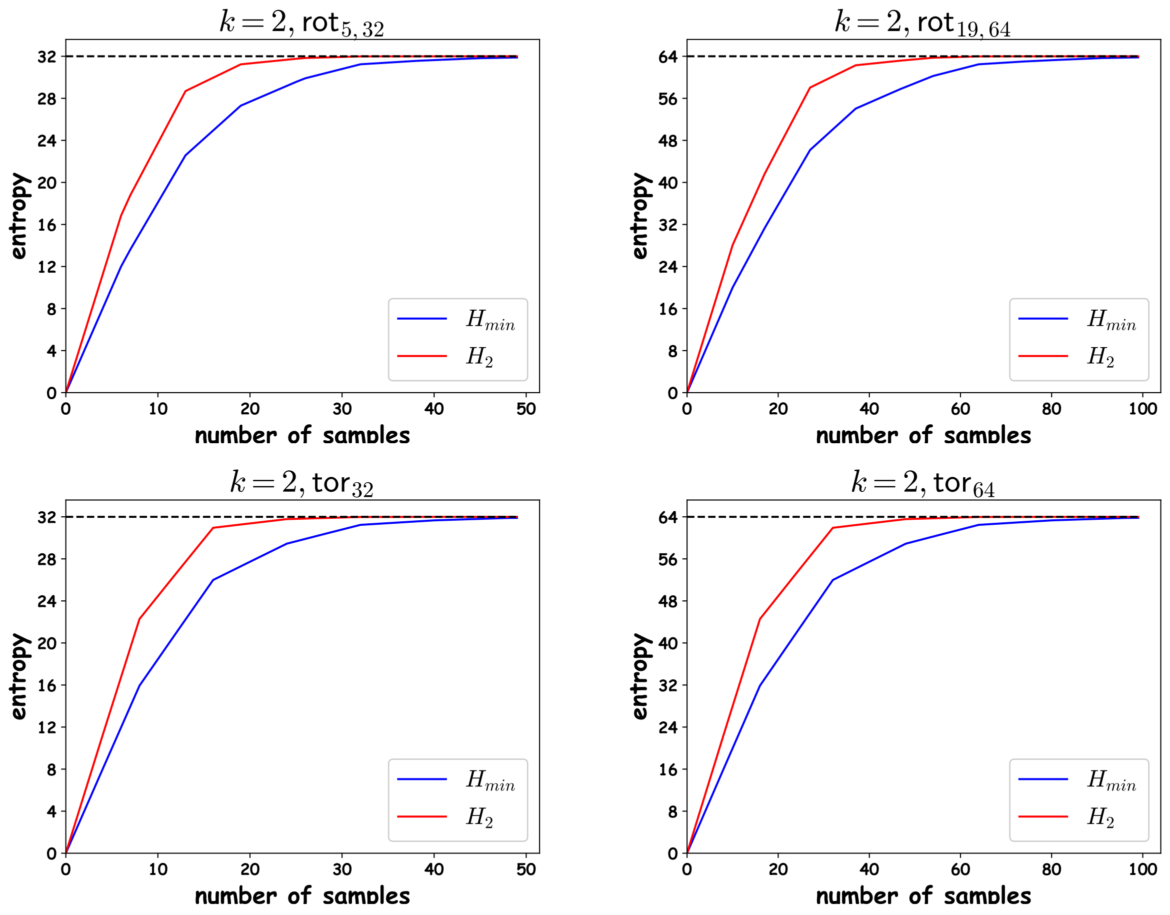

Figure 11: Number of bits of entropy (both H_2 and $H_{\min}$) accumulated by number of steps, started with an exponential distribution with two bits of min-entropy.

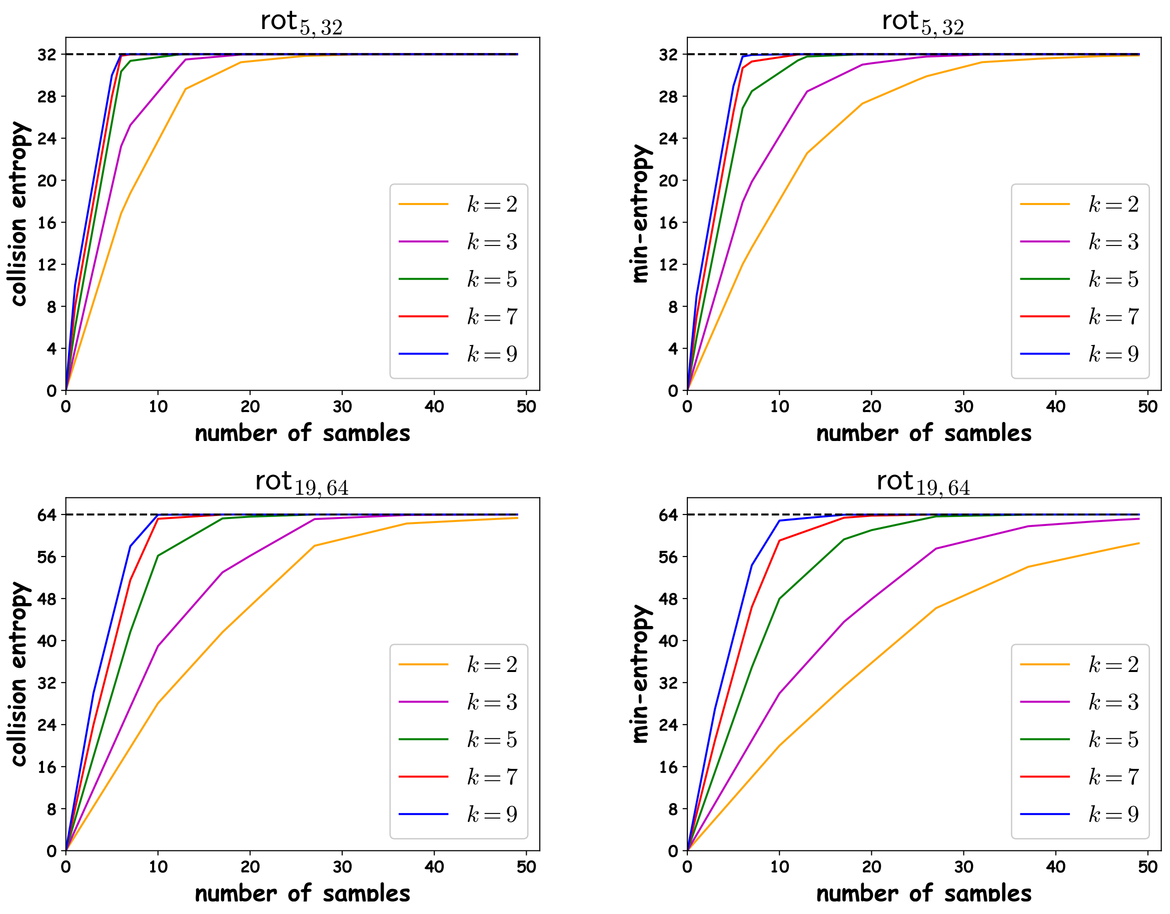

Figure 12: Number of bits of entropy (both H_2 and $H_{\min}$) accumulated by number of steps, starting with exponential distributions with different min-entropy. The two rotations shown are those chosen by Microsoft.

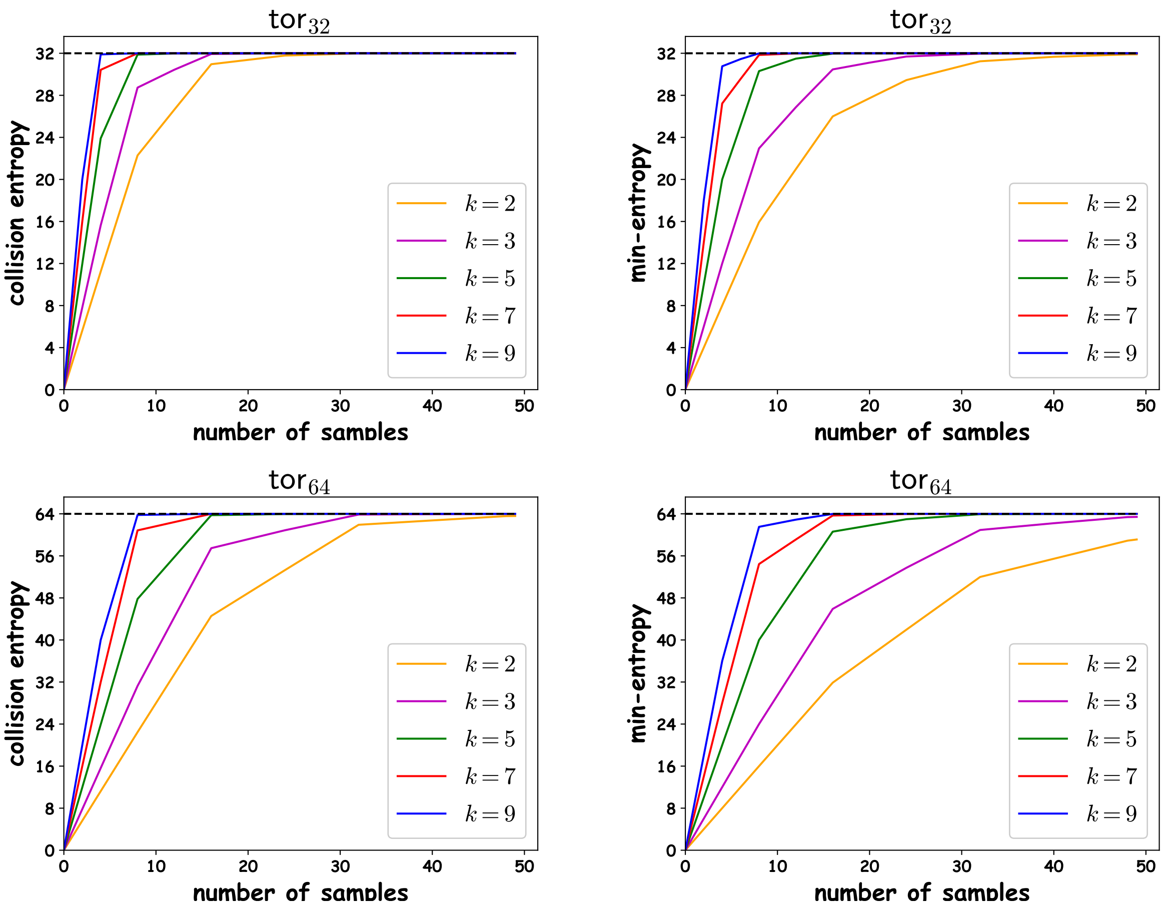

Figure 13: Number of bits of entropy (both H_2 and $H_{\min}$) accumulated using bit-reversed rotation by number of steps, starting with exponential distributions with different min-entropy.

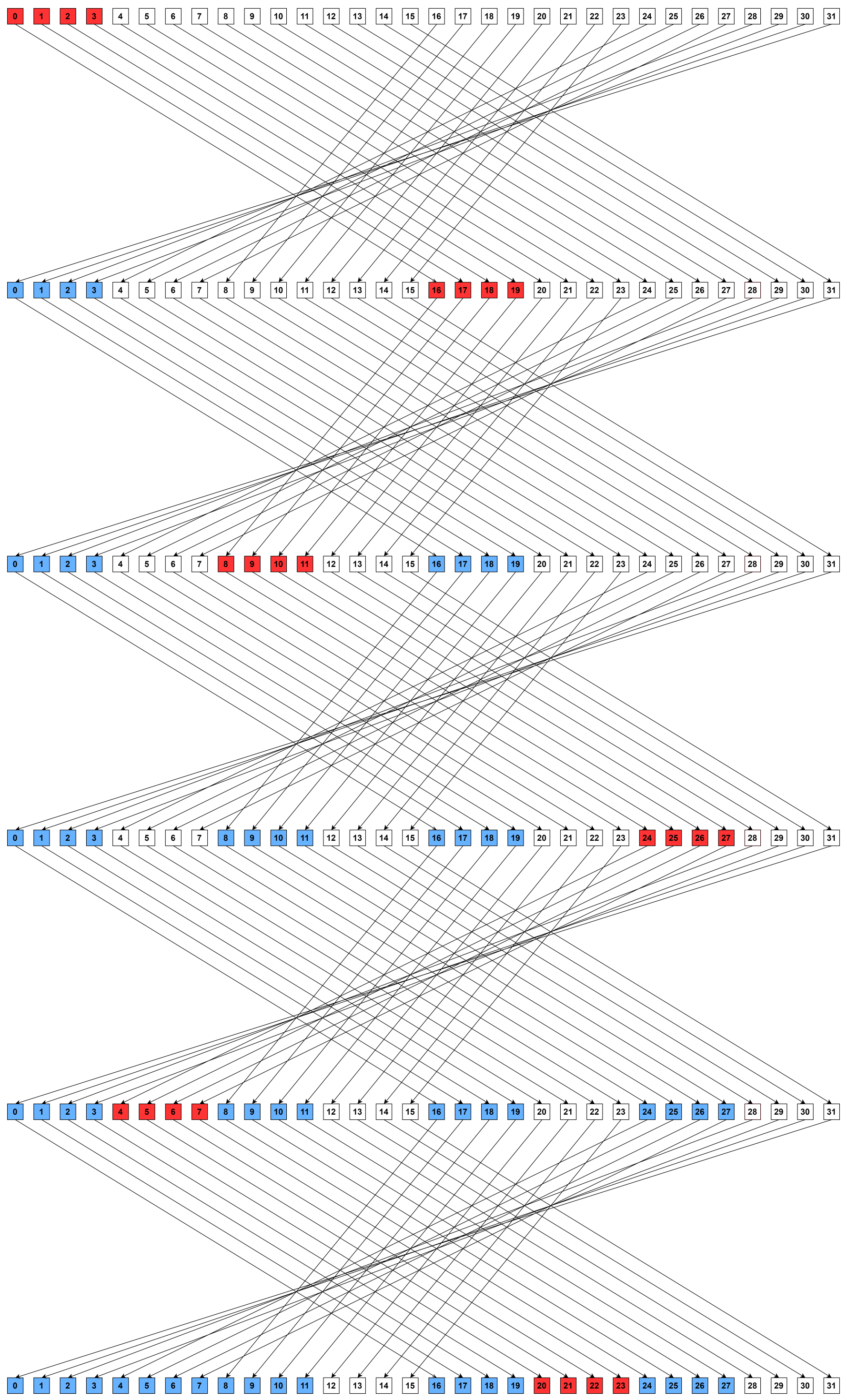

Figure 14: An illustration of bit-reversed rotation for $n = 32$. Red squares denote the position of lower-order bits (first four bits in our example) for different iterations of bit-reversed rotation. Blue squares denote positions that have been "covered" by these low-order bits.