Moonshine: An Online Randomness Distiller for Zero-Involvement Authentication

Jack West
Loyola University Chicago
jwest1@luc.edu

Kyuin Lee
University of Wisconsin–Madison
kyuin.lee@wisc.edu

Suman Banerjee
University of Wisconsin–Madison
suman@cs.wisc.edu

Younghyun Kim
University of Wisconsin–Madison
younghyun.kim@wisc.edu

George K. Thiruvathukal
Loyola University Chicago
gkt@cs.luc.edu

Neil Klingensmith
Loyola University Chicago
neil@cs.luc.edu

ABSTRACT

Context-based authentication is a method for transparently validating another device's legitimacy to join a network based on location. Devices can pair with one another by continuously harvesting environmental noise to generate a random key with no user involvement. However, there are gaps in our understanding of the theoretical limitations of environmental noise harvesting, making it difficult for researchers to build efficient algorithms for sampling environmental noise and distilling keys from that noise. This work explores the information-theoretic capacity of context-based authentication mechanisms to generate random bit strings from environmental noise sources with known properties. Using only mild assumptions about the source process's characteristics, we demonstrate that commonly-used bit extraction algorithms extract only about 10% of the available randomness from a source noise process. We present an efficient algorithm to improve the quality of keys generated by context-based methods and evaluate it on real key extraction hardware. Moonshine is a randomness distiller which is more efficient at extracting bits from an environmental entropy source than existing methods. Our techniques nearly double the quality of keys as measured by the NIST test suite, producing keys that can be used in real-world authentication scenarios.

IPSN '21, May 18–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8098-0/21/05…$15.00
https://doi.org/10.1145/3412382.3458899

CCS CONCEPTS

• **Security and privacy → Embedded systems security**.

KEYWORDS

Security and Privacy, Randomness Distillation

ACM Reference Format:
Jack West, Kyuin Lee, Suman Banerjee, Younghyun Kim, George K. Thiruvathukal, and Neil Klingensmith. 2021. Moonshine: An Online Randomness Distiller for Zero-Involvement Authentication. In *The 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021) (IPSN '21), May 18–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3412382.3458899

1 INTRODUCTION

Context-based authentication is emerging as a solution to enable fast and convenient device authentication. It verifies two devices' coexistence by comparing an independently generated random key or pin from an ambient source of randomness, such as wireless signal strength, acoustic noise, electrical noise, etc.

A pair of devices performing context-based authentication begin by individually harvesting noise from a shared source of randomness (Fig. 1). Each device samples the environmental signal with an analog-to-digital converter and divides its sequence of samples into blocks. Each block of the samples is then converted into a single bit by way of a bit extraction technique. If the random source has a significant fraction of common-mode noise shared between the two authenticating devices, then the bit sequences they extract are substantially similar. The underlying assumption of context-based authentication is that only devices which are physically near one another share enough common-mode noise to authenticate. Distant devices, which are assumed not to be legitimate authenticators, generate significantly different bit sequences from the contextual information that they can observe, so they cannot authenticate with a legitimate device.

The environment around us is full of noise sources, but all noise sources ultimately have the same problem: we need

some filter or transformation to translate samples of the noise source into the keyspace elements. The question we are trying to answer here is how do we efficiently build transformation functions that can produce high-quality keys.

As we will demonstrate, most environmental noise sources generate signals with a low *randomness density*. Informally, we say that if we harvest a k-bit key from an environmental source, that key could be represented with a shorter sequence of $k \quad n$ bits. More formally, we say that the Shannon entropy rate of a k-bit key generated from an environmental noise source is $H_e[X] < H_U[X]$, where $H_U[X] = k$ is the entropy rate of an iid sequence of k uniformly-distributed Bernoulli random variables.

One possible solution to this problem would be to use the bit sequence extracted from the environmental noise source as a seed for a pseudorandom number generator (PRNG), which would produce keys that are indistinguishable from random. Since PRNGs are deterministic, all devices that start with the same seed would produce the same key. Although, if we assume that an illegitimate user who wishes to gain access to a network that is secured by context-based authentication knows how the PRNG converts seeds to keys—an assumption we must make—then the keys produced by the PRNG can be of no higher quality than the seeds used to generate them. Using cryptographic hash functions to randomize a low-entropy bit sequence results in a similar problem.

In this work, we answer two open questions regarding the information theoretic properties of context-based key generation. First, we ask *what is the maximum amount of randomness we can extract (in bits per second) from some environmental source process given only its power spectral density?* Second, we ask *how can we increase the randomness density in keys generated by context-based methods?* In the existing literature, no technique can repeatedly generate shared keys from environmental noise that can pass standard tests for randomness—Table 1 shows a summary of NIST results for many recent pieces of work that report their results. Without generating sufficiently random keys, we cannot be sure that we are excluding unauthorized users.

To address the first problem, we develop a method for estimating the entropy rate—which is an upper bound on the bit extraction rate—of the raw environmental noise signal. The bit extraction rate matters because the most secure keys are long. RSA and DSA keys are between 2,048–4,096 bits in length, and the minimum length is getting longer all the time due to improvements in computational capacity available for brute force attacks. For context-based authentication techniques to be practical, they must be faster than manual techniques like typing in a password. But current context-based authentication schemes are slow for two reasons. First, the entropy rate of the environmental noise process—measured in bits per second—limits the speed. Second, the efficiency of standard key generation algorithms that extract bits from the chosen environmental noise processes tends to be low. Standard algorithms can extract one key bit every 10–20 samples of the noise process.

To address the second problem, we introduce a new randomness distiller called Moonshine. Moonshine distills randomness from a long bit sequence with low-entropy density into a shorter bit sequence with high entropy density that can be used as a secure cryptographic key. Moonshine produces keys with near-optimal entropy rate by selectively discarding subsequences from the input. By contrast, other randomness correctors have suboptimal performance [8].

The technical challenges of implementing a randomness distiller stem from the fact that random numbers' predictability is difficult to measure. There are many senses in which a bit sequence may be predictable: its periodicity, propensity to generate a particular bit sequence (even if that sequence's appearance is not periodic), etc. Furthermore, once we know that a bit sequence is predictable, it is not easy to selectively eliminate its predictable elements. Our theoretical analysis of environmental entropy sources suggests techniques to deal with these challenges and motivates our design of Moonshine.

This work introduces a novel randomness distiller in the data pipeline shown in Fig. 1 between bit extraction and key reconciliation which selectively removes bits generated by the bit extractor to improve the quality of the final key that is generated. Our randomness distiller introduces the new technique of discarding blocks of bits from the input sequence in a periodic fashion. This has the effect of disrupting the periodic repetitions on the input sequence, with the result being more random sequences that do well on the NIST benchmarks. We demonstrate that this bit discarding technique significantly improves the overall quality of keys while adding only a negligible processing overhead.

We demonstrate the effectiveness of Moonshine by evaluating it on real context-based authentication hardware. The authors of VoltKey lent us prototypes that we used to evaluate Moonshine. The Moonshine corrector improves the randomness of bit sequences generated by VoltKey and considerably outperforms the Von Neumann randomness corrector, both in terms of the randomness of the resulting key and the amount of data retained after correction. Keys generated by Moonshine can pass 14/15 NIST tests for randomness, making them suitable for use as cryptographic and authentication keys. It is important to know the source process's entropy rate because that imposes a hard upper limit on the speed of key extraction. We might choose to work with one noise process or another, depending on the entropy rate.

Figure 1: Data pipeline for VoltKey. Moonshine is applied during the bit extraction phase.

Generally speaking, when we are designing a context-based authentication scheme, we know the general properties of the source process's power spectral density. For example, body-area networks that harvest electrical signals from the heart (H2H [24] and H2B [18]) to generate authentication keys generally start with an analog source process that has a bandwidth of a few Hertz. We can assume that its power spectral density will have a few strong harmonics in the 1 10 Hz frequency band superimposed on additive white Gaussian noise. VoltKey, which harvests randomness from the electric power lines, will likely have strong harmonics at multiples of 60 Hz, tapering o at a few hundred Hertz. Starting with this information, we show how to estimate the entropy rate of a typical realization of the source process and improve the quality of extracted keys.

The contributions of this work are as follows:

(1) We present a new method for calculating the entropy rate of a random process from its samples or its PSD that works for any wide-sense stationary random process (Ÿ3.3 & Ÿ3.4).
(2) Using intermediate results from the previous technique, we introduce Moonshine, an entropy distiller that can achieve a substantially improved pass rate of the NIST test for randomness.
(3) We preformed a comparative analysis of key quality from di erent bit streams with di erent bit extraction algorithms and environmental sources.
(4) We build a prototype implementation of Moonshine that runs on real context-based authentication hardware. We evaluate our prototype on several types of environmental noise. Our results show that Moonshine is generalizable to di erent context based authentication mechanisms.

Our methods make some mild assumptions about the characteristics of the environmental source noise process. We model the source process as the sum of a band-limited deterministic signal and additive white Gaussian noise.

2 BACKGROUND

Devices that use context-based security take advantage of the fact that the common contextual information is shared only by a limited group of closely located devices. The presence of common contextual information is evidence that the devices are located in the same place simultaneously, which implies that they legitimately belong to the same user. The keys generated from contextual information can establish initial trust (as a pairing key) and protect subsequent communication (as a cryptographic key). This eliminates the need for human involvement for making, entering, and managing a secret key, which can dramatically improve the overall usability of systems that currently rely on passwords to protect data. In addition, the time-varying nature of contextual information also allows devices to use a new key for each pairing attempt or periodically update the cryptographic key, which signi - cantly reduces the attack window for adversarial agents.

Similar studies [18, 19, 21, 31], show that the act of calculating the amount of randomness in an environmental signal is not straightforward. We cannot just sample the source and create a histogram to compute the statistical entropy because the samples are not independent. Most of the time, environmental noise contains some deterministic component, causing samples to be correlated. This deterministic component makes the entropy computation meaningless. Techniques exist for computing entropy rates from a signal's power spectral density (PSD), but they are unstable for correlated random processes. Our contribution is to introduce a stable method to calculate the entropy rate from a signal's PSD that works on arbitrary random processes.

2.1 An Overview of the Standard Context Based Authentication Process

This section gives a broad overview of the process devices and goes through to generate random keys from environmental noise. We assume that there are two devices that are both located near each other and measuring the same random environmental noise. Most context-based authentication schemes involve three basic steps: noise harvesting, key generation, and reconciliation.

(1) Noise Harvesting: In the rst step, the device gathers a sequence of samples from an environmental noise process. This is usually done by a microcontroller with an analog-to-digital converter. These samples are typically ltered to

remove undesirable features and time synchronized by sending messages over a public channel.
(2) Bit Extraction (the focus of this paper): Raw samples gathered from the environmental noise process are then converted by a fuzzy extractor into a sequence of bits that will be a key. Generally, a few bit errors (1-10%) between authenticating devices are permitted in the extracted bit sequence. The most popular bit extraction technique is to divide the sequence of raw noise samples into bins of 10-20 samples each.
(3) Key Reconciliation: After bit extraction, each device will have a sequence of bits in memory that represents a key. Even though the devices are located nearby one another, di erences in their measurements of the raw environmental noise will have caused spurious errors in the extracted bit sequences. Key reconciliation is the process by which two nearby devices exchange messages with one another over a public channel to resolve those bit di erences. At the end of this step, if both devices are honest and located in the same vicinity, they will each hold identical authentication keys that can be used to encrypt data or validate their identities to one another.

Fuzzy Extractors. Context-based authentication relies on the authenticating devices observing nearly identical environmental noise signals to base their keys. But even two nearby devices may read di erent values from their respective sensors during an event that creates noise. If the event is closer to one device than the other, they will observe slightly di erent noise patterns and generate di erent keys. Fuzzy extractors account for inconsistencies in observed environmental noise by examining noise and use a mapping function, which all the devices in the system share, to map the observed noise to a new value.

NIST Test for Randomness. The NIST test for randomness is a software suite that evaluates the quality of a random bit sequence. It consists of 15 separate tests that analyze the bit stream bit-wise, block-wise, and superblock-wise. Moonshine's goal is to modify a bit sequence to increase the number of NIST tests that it passes. Passing more NIST tests should be the goal of any random number generator to verify that generated keys are random. We use the NIST tests for randomness as a benchmark for key quality.

The NIST test suite's input is a bit sequence, typically thousands to hundreds of thousands of bits in length. For each test in the suite, the input bit sequence is divided into blocks, and each block is evaluated independently. The output of a typical run of the suite, shown in Table 1, lists two crucial gures for each of the 15 tests: a p-value, and the proportion of bits, blocks, or superblocks that passed the test.

p is the probability that a true random number generator would have produced a less random output than the given test input sequence. p values closer to 1 are better, and p values closer to 0 are worse. $C_1, C_2, \ldots C_{10}$ represent the number of p values that lie in the intervals $[0.0, 0.1)$, $[0.1, 0.2)$, $\ldots$, $[0.9, 1.0]$.

Figure 2: A sorted histogram of variable-length bit sequences generated from measured environmental noise.

The proportion output gives the fraction of blocks that passed each test in the suite. We want the proportion of passes to be as close to unity as possible. In general, the tests in the NIST suite need a minimum of 1000 bits to evaluate the quality of the sequence shorter sequences cannot be evaluated with high con dence. Each stream of bits needs to be at least 1000 bits, or our NIST test suite fails internally. Therefore, if enough bits exist, we divide the stream into 100 blocks of at least 1000 bits in size. The test suite divides sequences into blocks of 100 bits each and subjects each block to the suite of 15 tests.

2.2 A Review of the Typical Set and the Asymptotic Equipartition Property

One of the most signi cant characteristics of cryptographic keys is that their bit sequences be uniformly distributed that is that each bit sequence should be equally probable. If some bit sequences in the key are more likely than others, it would be easy for an attacker to guess the chosen key. But sources that generate independent and uniformly distributed random numbers are typically challenging to build. As previous work in context-based authentication has demonstrated, most environmental noise does not yield uniformly distributed samples . How can we generate uniformly distributed numbers from a nonuniform source?

Suppose we independently sample a nonuniform source many times in succession. The sequences of samples we obtain from that sampling process can be divided into a typical set and a non-typical set. The probability that a sequence of independent samples lies in the typical set approaches 1 for su ciently long sequences. Furthermore, all sequences in the typical set are almost uniformly distributed. This result is called the asymptotic equipartition property. Figure 2 shows

histograms of bit sequences of di erent lengths, sorted from most probable to least probable. Bit sequences do not need to be very long before they begin to exhibit this almost-uniform characteristic.

2.3 Rényi Entropy

De nition 2.1. Rényi Entropy Let X be a random variable with an alphabet $\mathcal{X}$ and distribution $p_X(x)$. The Rényi Entropy of X is de ned as

$$R(X) = \log_2 \tilde{O}_{x \in \mathcal{X}} p_X(x)^2$$

Bennett et. al. demonstrated [2] that the Rényi entropy is a lower bound on the number of bits of private information that can be distilled after key reconciliation. More generally, Rényi entropy is a convenient tool for estimating the uncertainty in a random variable. By Jensen's inequality, we have that Rényi entropy is upper bounded by the Shannon entropy:

$$R(X) \leq H(X)$$

with equality when $X \sim Uniform(X)$.

2.4 Related Work

The literature that is adjacent to our work can be roughly separated into two categories: context-based authentication systems and cryptography.

Context-Based Authentication. This body of work includes full-system implementations of context-based authentication systems. In general, each system includes a mechanism for measuring environmental randomness, extracting bit sequences, and resolving bit errors in the bit sequences to form identical keys. Many di erent sources of environmental randomness have been studied. For body area networks of wearable devices, the H2H [24], H2B [18], and others [1, 37] systems measure ECG signals (heartbeat data). Secret From Muscle [36] EMG (produced by skeletal muscles) and skin vibration has been used to generate keys between low-cost wearable devices and implantable medical devices. Context-based authentication systems targeted at stationary IoT devices have used context from audio, humidity, luminosity, visual, and vibration channels [17, 22, 23, 26, 27, 29]. ProxiMate, Amigo, Ensemble, and others [7, 12, 15, 19, 32] extract entropy from measurements of the radio frequency spectrum. Because the randomness density of environmental signals tends to be low, context-based authentication systems generally pass half or less than half of the NIST tests. We have compiled the results of the NIST tests from a subset of the context-based authentication systems in Table 1[1].

Cryptography. In the domain of cryptography and information theory, a lot of work has focused on understanding the information content of signals. Key reconciliation a suite of techniques that context-based authentication relies heavily on was originally developed for exchanging information over quantum communication channels [2, 3, 20]. Fuzzy extractors [5], fuzzy vaults [13], fuzzy commitment [14], and fPAKEs [6] are more modern cryptographic techniques that are commonly used in key reconciliation by context-based authentication systems. Also other authors have built techniques to create a more uniform distribution of biased random number generators [8].

3 ENTROPY RATE OF A NOISY BANDLIMITED SIGNAL

This section develops a new technique for measuring the entropy rate of a bandlimited signal from its power spectral density (PSD). We begin with the Burg Max Entropy Theorem, which compares a signal's PSD and its entropy rate. But as we will see, Burg's theorem is intractable to compute for signals more than a few samples in length. We develop a computationally-tractable method for calculating the entropy rate of reasonably-sized signals (several thousand samples long).

We could use the Shannon-Hartley Channel Capacity theorem, which relates the noisy signal's entropy rate to an SNR. The di culty with this method is that it is not clear how to interpret the SNR in situations where we are dealing only with noise. In most context-based authentication scenarios, we want as much uncorrelated noise as possible, and we try to lter out everything else. We would prefer to have an expression for entropy rate that is a function of the power spectral density of the harvested randomness signal because that does not require us to measure the SNR.

3.1 System Model

Suppose we harvest entropy from some environmental signal that is composed of a deterministic part and a random part:

$$X(t) = D(t) + Z(t) \quad (1)$$

Where $D(t)$ is an unknown bandlimited signal deterministic in time and $Z(t)$ is additive white Gaussian noise (AWGN). $X(t)$, $D(t)$ and $Z(t)$ are assumed to be continuous in time. This model is used by many context-based authentication schemes, including VoltKey [16], H2H and H2B [18, 24]. We can represent $D(t)$ as an expansion in the basis of sinusoids:

[1] Not all context-based authentication systems publish their NIST test results. The results shown in Table 1 are all the published results that we could nd.

Table 1: NIST test results for various context-based authentication schemes (X indicates pass).

	Pass Rate	Frequency	Block Frequency	Cumul. Sums, Fwd	Cumul. Sums, Rev	Runs	Longest Run	Rank	FFT	Non-overlap Template	Overlap Template	Universal	Approx Entropy	Serial	Serial	Linear Complex
Jana et al. [12]	10/15	X	X	X	X	X	X		X				X	X	X	
H2B [18]	8/15	X	X	X			X		X	X			X			X
H2H [24]	8/15	X				X	X	X	X			X	X			X
Xi et al. [35]	10/15	X	X	X	X	X	X		X				X	X	X	
Secret from Muscle [36]	9/15	X	X	X	X	X	X		X				X	X		
VoltKey [16]	8/15	X	X	X	X			X		X			X			X
VoltKey + Von Neumann corr.	11/15	X	X	X	X			X	X	X			X	X	X	X
VoltKey + Moonshine	14/15	X	X	X	X	X	X	X	X	X	X		X	X	X	X

$$D(t) = \sum_{k=0}^{N} a_k \cos(2\pi f_k t + \theta_k)$$

Where the a_ks are expansion coefficients and θ_k is the phase angle, modelled as a uniformly distributed random variable on $[0, 2\pi)$.

Theorem 3.1. $D(t)$ is a stationary process.

Proof. See Appendix A

The autocorrelation function is the same for all time shifts, meaning that its value is only dependent on the lag $t_2 - t_1$, not on the absolute time t_1 or t_2. We have verified that the ACF depends only on time shift for functions of the same form with (a) more than two terms and (b) various combinations of coefficients on each term. This is the criterion for stationarity.

We model the entropy source $X(t)$ as a stochastic process that can be sampled discretely in time, yielding a sequence $\{X_m\}$. In this paper, $\{X_m\}$ is the sequence of ADC samples of the environmental noise process .

3.2 Computing the Entropy Rate from the PSD

Our goal is to get a bound on the source process's entropy rate $\{X_i\}$, which can be directly computed from the process's power spectral density. We want an inequality similar to the Shannon-Hartley channel capacity bound but where the signal bandwidth and SNR are directly computed from the properties of the source process's PSD.

Assuming that the entropy source is a wide-sense stationary (WSS) stochastic process (as we demonstrated in Theorem 3.1), we can compute its autocorrelation function by taking the inverse Fourier transform of the PSD[2]:

$$R_{XX}(\tau) = \mathcal{F}^{-1}\{S(f)\}(\tau).$$

[2]This is called the Wiener Khinchin theorem.

Theorem 3.2. Burg's Maximum Entropy Theorem

In general, the maximum entropy rate stochastic process $\{X_i\}$ satisfying the constraints

$$R_{XX}(k) = E[X_m X_{m+k}] = \alpha_k,\ k = 0, 1, 2, \ldots, p \qquad (2)$$

is the pth-order Gauss-Markov process of the form

$$X_m = \sum_{k=1}^{p} a_k X_{m-k} + Z_m \qquad (3)$$

Proof. See [4].

In other words, each new sample X_m is a linear combination of the previous p samples plus some iid additive white Gaussian noise $Z_m \sim N(0, \sigma^2)$. The a_ks are chosen to satisfy Equation 2. We can use the Yule-Walker equations to calculate the a_ks from and the α_ks, the values of the autocorrelation function of the source process $\{X_i\}$ [4].

Burg's maximum entropy theorem holds without assuming that $\{X_m\}$ is broad sense stationary. However, to compute the α_ks from the PSD, we need the source process $D(t)$ to be wide sense stationary. We can write each X_{m-k} as an expansion in the basis of complex sinusoids:

$$X_m = \sum_{k=1}^{N} \sum_{l=0}^{N} a_k b_l e^{i 2\pi f_l (m-k)} + Z_m \qquad (4)$$

In Theorem 3.1, we demonstrated that the source process of interest is stationary under some reasonable assumptions, and therefore it satisfies the conditions in Equation 2. In fact, the α_ks from Equation 2 can be directly read from the autocorrelation function of the source process, which is the inverse Fourier transform of the PSD.

3.3 Computing Rényi Entropy Rate

Plugging in our model for X_m from Equation 4 into the expression for Rényi Entropy:

$$R(X_m) = \log_2 E\left[p_X(x)^2\right] = \log_2 \sum_{x \in \mathcal{X}} \Pr\left[\sum_{k=1}^{p} \sum_{l=0}^{p} a_k b_l e^{\frac{i2\pi lm}{N}} + Z_m = x\right]$$

Where in the second step, we plug in the discrete Fourier transform representation of our signal. We will now modify the limits on the double sum. First, we will set $p = N$, so that the Markov process's order is a complete sampling window. Second, we will drop the $l = 0$ term in the DFT, assuming no DC offset, that the sample process has zero mean. This allows us to combine the double sum into one sum that runs from 1 to N.

$$R(X_m) = \log_2 \sum_{x=0}^{2^b - 1} \Pr\left[\sum_{k=1}^{N} a_k b_k e^{\frac{i2\pi km}{N}} + Z_m = x\right]$$

Here, the $a_k = \sqrt{P_k}$, the samples of the PSD of the deterministic signal. The outer sum over x runs from 0 to $2^b - 1$ because we are assuming that the signal is being acquired with a b-bit analog to digital converter.

3.3.1 If Deterministic Signal is Small. Suppose the deterministic component of the signal, $D(t)$ is zero, and the only source of entropy is Z_m, the additive white Gaussian noise, which we assume is correlated among two or more context-based authenticators. For example, this would be the case in the H2H body area network when there is no ECG signal on the skin, or in VoltKey when there is no 120VAC power waveform. The only noise present is caused by cosmic radiation.

This is a slightly unrealistic assumption, but it allows us to comprehend the amount of entropy carried by the correlated random noise, whose statistical properties we know. In this calculation, we want to find the amount of entropy carried by the AWGN, not the amount of shared entropy called mutual information common to both devices.

Theorem 3.3. *The Rényi entropy of the samples of the source process $\{X_i\}$ is lower bounded by the Rényi entropy of the AWGN Z_m. In other words, $R(X) \geq R(Z)$. After observing the deterministic component of the source process, the remaining uncertainty in X is due to Z, the AWGN.*

Proof. See Appendix B.

Theorem 3.4. *Let Z_m be a single sample of the analog source process $\{X_i\}$ consisting only of additive white Gaussian noise. Then its Rényi entropy is $R(Z_m) = \log_2 2\sqrt{\pi}$.*

Proof.

$$R(X_m) = \log_2 \sum_{k=-2^{b-1}}^{2^{b-1}-1} \Pr\left[b_k \leq Z_m < b_{k+1}\right]^2$$

$$\log_2 \sum_{k=-2^{b-1}}^{2^{b-1}-1} \frac{1}{\sigma\sqrt{2\pi}} e^{-k^2/2\sigma^2}$$

$$\log_2 \int_{-2^{b-1}}^{2^{b-1}+1} \frac{1}{\sigma\sqrt{2\pi}} e^{-k^2/2\sigma^2} dk$$

$$\log_2 2\sqrt{\pi}$$

Here we assume that the analog AWGN signal will be quantized by an analog-to-digital converter with b bits of precision. The sum over the alphabet $\mathcal{X}$ is over all of the 2^b possible quantizations that can be produced by a b-bit ADC. In the last approximation, we are assuming that σ is small, and we are capturing most of the probability of Z_m within the limits of the ADC's dynamic range.

In a typical application where we are intentionally amplifying correlated random noise Z_m, we can expect σ to be on the order of $1 \cdot 10^{-3}$ of the dynamic range of our ADC. For reasonable values of σ, the Rényi entropy is limited to about 10 12 bits per sample.

This is a lower bound on the Shannon entropy of $X(t)$ for two reasons. First, as discussed in Ÿ2.3, $R(X) \leq H(X)$ for all random variables X. Second, our computations in this section assume that the deterministic component of $X(t)$ is zero. Adding a nonzero deterministic component will increase the entropy per sample (for proof of this claim, see Appendix B).

Still, the Rényi entropy rate of Z_m is relatively high, even if we ignore the deterministic component of the signal. Standard key extraction techniques used by context-based authentication mechanisms are only able to extract one bit of entropy per 10 ADC samples: a bit extraction rate about two orders of magnitude lower than what we would expect to be carried by AWGN noise process alone!

3.3.2 If Deterministic Signal is Nonzero. In the case where pth order Gauss-Markov process, it is possible to compute the entropy rate without the Yule-Walker equations.

$$H(X) = H(X_p | X_{p-1}, \ldots, X_0) \quad (5)$$

$$= H(X_0, \ldots, X_p) - H(X_0, \ldots, X_{p-1}) \quad (6)$$

$$= \frac{1}{2}\log_2(2\pi e)^{p+1}|K_p| - \frac{1}{2}\log_2(2\pi e)^{p}|K_{p-1}| \quad (7)$$

$$= \frac{1}{2}\log_2(2\pi e)\frac{|K_p|}{|K_{p-1}|} \quad (8)$$

where K_p is the autocorrelation matrix of the process $\{X_m\}$. The K_p autocorrelation matrix is called a Toeplitz matrix. It is rank N. Its top row is the individual values of the autocorrelation function of the source process $\{X_k\}$. Using the fact that

$$E[X_k X_l] = R_{XX}(k - l) = R_{XX}(l - k)$$

for wide-sense stationary processes:

$$= \begin{bmatrix} R_{XX}(0) & R_{XX}(1) & R_{XX}(2) & \cdots & R_{XX}(N-1) \\ R_{XX}(1) & R_{XX}(0) & R_{XX}(1) & \cdots & R_{XX}(N-2) \\ R_{XX}(2) & R_{XX}(1) & R_{XX}(0) & \cdots & R_{XX}(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{XX}(N-1) & R_{XX}(N-2) & R_{XX}(N-3) & \cdots & R_{XX}(0) \end{bmatrix}$$

Equation 8 gives the entropy rate in terms of the source process's autocorrelation function. Unfortunately, Equation 8 presents a fairly serious problem: the above result cannot be computed directly for large values of N because the determinant of the autocorrelation matrix approaches zero as N increases. Computing the determinant of large matrices is a problem in general and not a pathology that is isolated to our autocorrelation matrix. We discuss below a method for computing the ratio $|K_p| \div |K_{p-1}|$ without directly computing the determinants.

3.4 Calculating Entropy Rate without Directly Computing Determinants

The problem that we encounter when attempting to compute an entropy rate from Equation 8 is that the columns of the autocorrelation matrix K_p are almost linearly dependent. This makes the determinants of K_p and K_{p-1} close to but not exactly zero, and the computer's floating point representation rounds the result of the determinant computation to zero. Our goal here is to compute the ratio of the determinants, so we are not concerned about the fact that they are individaully small.

Lemma 3.1. If R is an $N \times N$ triangular matrix, then the determinant of R is the product of its diagonal elements:

$$|R| = \prod_{i=1}^{N} r_{ii}$$

$|K_p|$ and $|R|$ are close to zero because all of its diagonal elements are less than 1, causing the product in Lemma 3.1 to approach zero for large N. To be clear, R and K_p are both full-rank matrices.

Lemma 3.2. If $A = Q_1R_1 = Q_2R_2$ are two QR decompositions of full rank square matrix A, then

$$Q_2 = Q_1S$$
$$R_2 = SR_1$$

for some square diagonal S with entries ± 1. If we require the diagonal elements of R to be positive, then the factorization is unique.

Proof. Starting with the factorization $A = Q_1R_1 = Q_2R_2$ we can define a new matrix S in the following way:

$$S = Q_2^T Q_1 = R_2 R_1^{-1}$$

Since Q_1 and Q_2 are unitary, then S must also be unitary. Since R_1 and R_2^{-1} are upper triangular, then S must also be upper triangular. This means that S is a diagonal matrix with elements ± 1.

$$Q_1R_1 = Q_2R_2$$

$$Q_1R_1R_2^{-1} = Q_1S = Q_2$$

Now, apply the constraint that R_1 and R_2 have positive entries on their diagonals, which forces $S = I$ and $R_1 = R_2$.

Theorem 3.5. The ratio $|K_p| \div |K_{p-1}| = r_{pp}$, the lower rightmost element of R in the QR factorization of K_p.

Proof.

$$\prod_{k=1}^{p} r_{kk} = 10^{\log \prod_{k=1}^{p} r_{kk}} = 10^{\sum_{k=1}^{p} \log r_{kk}}$$

$$\frac{|K_p|}{|K_{p-1}|} = \frac{10^{\sum_{k=1}^{p} \log r_{kk}}}{10^{\sum_{k=1}^{p-1} \log r_{kk}}} = 10^{\sum_{k=1}^{p} \log r_{kk} - \sum_{k=1}^{p-1} \log r_{kk}} = r_{pp}$$

Note that there is a unique QR factorization of K_p that has all positive diagonal elements of R. We need R to have positive elements on its diagonal in order to be able to take their logarithm. Using the above technique for computing the ratio $|K_p| \div |K_{p-1}|$, we can rewrite Equation 8 by bringing the factor of $1 \div 2$ into the logarithm and assuming that the source process consists only of uncorrelated Gaussian noise:

$$H(X) = \log_2 \sqrt{2\,e}$$

Where we replace the ratio of determinants with the standard deviation of the source process. The Shannon entropy differs from the Rényi entropy only by a factor of $\sqrt{e \div 2}$.

3.5 Measuring Shared Randomness

The idea that underlies context-based authentication techniques is that two devices can build a shared key from a shared randomness source, which is not observable by third parties. Evaluating the key generation scheme matters not just how much randomness is encoded in the signal to begin with but how much of that randomness is common to both devices.

The challenge in doing a theoretical analysis of the amount of mutual information that we would expect to be common to two context-based authenticators is that it tends to be situationally dependent. Di erent environments, di erent physical con gurations of devices, and other parameters weigh heavily on the correlation between two source processes.

To compare the amount of mutual information between two environmental signals to the amount of entropy, we collected some voltage measurements at high frequency from the power outlets in our institution's o ces using VoltKey prototypes and analyzed their characteristics. We used standard techniques from Ÿ2.1 to sample and time-align voltage measurements without key extraction or reconciliation. We split each signal into blocks of 150,000 samples (about two seconds of data) and computed each block's mutual information and entropy.

There is an order of magnitude di erence in entropy and mutual information the amount of entropy common to two authenticating devices for the signal that we tested in this experiment. Said another way, only about 10% of the randomness measured by the VoltKey s is common to both devices. The entropy that is not common to both devices cannot be used to generate a key.

The entropy rate of the source process is about 16 bits per sample slightly higher than we would expect if we were gathering uncorrelated Gaussian noise. In Theorem 3.4, we concluded that we would expect roughly 10 11 bits of entropy per sample.

Based on our experience working with context-based authentication methods, we think that this order-of-magnitude-gap between entropy and mutual information is probably typical of many noise types. However, it is not easy to be con dent without conducting a more formal analysis.

What is surprising is that conventional algorithms for extracting keys from a natural environmental noise process only generate bits at about 1•10th the rate of the mutual information in this experiment. We should expect the bit extraction rates to be slightly lower than the mutual information rate to avoid errors perhaps 1•2 or 1•3 bit per sample but this discrepancy is ine cient. Even with such low bit extraction rates, most context-based authentication schemes still have to perform key reconciliation to eliminate bit errors in the shared key.

As we will see, the raw environmental signal's entropy rate a ects the performance of the randomness distiller.

4 RANDOMNESS DISTILLATION

Once we have extracted a bit sequence from an environmental source noise process, we need to distill its entropy to pass standard tests for randomness. Raw bit sequences extracted from the environment often do not pass standard randomness tests (Table 1).

Moonshine is a randomness corrector that transforms independent identically distributed samples from a random source to make their distribution closer to uniform. It works by concatenating samples of the source into sequences. By the asymptotic equipartition property, sequences of samples will be nearly uniformly distributed even if the individual samples are not. Von Neumann's corrector is a classical technique that aims to accomplish the same goal. We compared the preformance of Moonshine to Von Neumann's algorithm.

Von Neumann Corrector. The Von Neumann Corrector was an early technique used to normalize the histogram of randomly generated bit sequences [31]. Its goal is to generate bit sequences in which 1 and 0 are equally likely to occur from an input sequence of unfair coin tosses. The Von Neumann Corrector groups the input sequence into pairs of bits, and it discards pairs in which both bits are the same (f1; 1g and f0; 0g). For pairs of bits that are not the same, The Von Neumann Corrector copies only the first bit in the sequence to the output. It is essentially an application of a special case of the Asymptotic Equipartition Property, for which the sequence length is 2. The Von Neumann corrector did not improve the NIST test pass rate when we applied it to the raw VoltKey bit sequences (see Table 1) enough to use its output as a cryptographic key. We need a new technique to improve the quality of keys generated by context-based authentication systems.

Moonshine is a new entropy distillation technique that converts long bit sequences with low entropy per bit into shorter bit sequences with high entropy. Moonshine takes as input a bit sequence from a random generator and groups bits into blocks of k. Each block of k bits represents an integer in the range $[0; 2^k - 1]$. Moonshine then generates a histogram of all k-bit integers obtained from the raw input sequence. The resulting histogram is divided into two categories: the typical set, which is almost uniformly distributed, and the non-typical set, which occurs either much more frequently or much less frequently than average. Half of the input sequences are assumed to be part of the typical set, which is retained, and the other half of the sequences are discarded.

After discarding the non-typical set, we can represent the remaining elements of the typical set using $k - 1$ bit sequences. The new $k - 1$ bit indices are the output of the corrector. Fig. 3(b) shows the histogram of the raw bits extracted from a VoltKey device grouped into 7-bit blocks and assigned indices in the range $[0; 127]$. Fig. 3(a) shows the histogram of data after it has been corrected by Moonshine. For comparison, Fig. 3(c) shows a histogram of 7-bit numbers generated by python's random number generator.

In Fig. 4(a), we plot the number of bits remaining after correction by Moonshine as a function of the sequence length. The Von Neumann corrector discards over 80% of the bits in the input sequence. By tuning the subsequence length, Moonshine's bit usage can be adapted into any size of data given by the user.

4.1 Our Implementation of Moonshine

Moonshine examines a binary stream, partitions that stream into bit sequences, and creates a histogram of the partitioned bit sequences. Moonshine separates the extracted sequences into two categories: typical sequences and non-typical sequences. It then discards non-typical sequences and maps the typical sequences to new values based on those new histograms. The key components of our algorithm are:

(1) Sequences of size k get remapped into sequences of size $k - 1$.
(2) We drop bits after each individual sequence from the bit stream. We discuss the reasons for doing this at the end of this section.
(3) Bit sequences that occur most frequently on the input stream are dropped. The remaining sequences (the typical set) are remapped to new bit sequences.

The details of our algorithm are below.

(1) Partition Bit Stream into Sequences Moonshine is given a bit stream, while a system is running, generated from environmental noise as input. We first want to partition the bit stream into subsequences of length k. If we let $k = 8$, we then initialize four arrays two of them are of size 2^k while the other two arrays are size 2^{k-1}. These arrays each have different jobs. One of the arrays (called A) keeps track of the number of occurrences of each subsequence in the input bit stream. A holds a histogram of integers extracted from the environmental noise source, similar to that shown in Fig. 3. As we read bits from the input stream, we use array B to keep track of the order in which particular subsequences first occur. After converting a subsequence from the input of length k to an integer and updating arrays A and B, we skip m bits in the input sequence.
(2) Find Highest Occurring Half of Sequences To extract the typical set, we throw away the most commonly occurring subsequences of bits. To do that we compare the number of occurrences of each subsequence to all others in A. After knowing which subseqeuences are most common, we drop the most common. If a index is found to be in the highest half we change that index in A to be 1. We mark the same indices in B.
(3) Remap Binary Sequences and Return the Changed Stream The last array, C, will be used to hold the remappings. Now that we know which values to keep, we iterate through B and if the value is not marked as 1 remap it to the next available integer in the C. If a number appears in the stream a second time that hasn't been remapped too, we throw it out. After we preform these steps the result will be a bit extracted stream. We also skip over the same amount of bits we did in the first step. For example, if 192 is the first number that appears in the bit stream, and when indexed in array A the value is not 1. Whenever we see the value 192 in the bit stream, we replace that value with its remapped equivalent that is defined in C. So, every occurrence of 192 in the input data stream will be replaced with zero. Also, we are replacing an original bit stream of size k with a new bit stream of size $k - 1$. Therefore dropping the last bit.

Real Environmental Signals Our analysis in §3.4 assumes that the environmental signal is a stationary process, which is a valid assumption during periods of everyday activity. But during periods of little or no activity, many environmental noise signals are not stationary. During inactivity, the entropy rate is effectively zero, causing samples of the source noise process not to be independent or identically distributed. The result is that we get very long runs of zeros or ones in our extracted bit sequences during periods of inactivity. We can repair the bit sequence by dropping bits from the input.

Bit dropping has the effect of shortening long runs of ones or zeros during periods of inactivity. It can not make a non-stationary process stationary, but it can reduce the length of non-stationary sequences, making them less influential in the overall signal's properties.

An alternative we considered is to wait until the source noise process has a high enough entropy rate before beginning key generation. But waiting for the environmental entropy rate to increase above a threshold is impractical for real systems because two authenticating devices may not agree on the exact moment when the entropy rate becomes high enough. Coordinating between multiple devices would require additional communication, which wastes time. Another advantage of bit dropping is that it allows keys to be generated immediately when requested by the user rather than waiting for an acceptable entropy rate in the environmental noise.

4.2 Choosing Parameters k and m

The quality of keys generated by Moonshine depends heavily on our parameters in Algorithm 1. The relationship between parameter values and key quality is generally monotonic higher parameter values usually produce better keys. Larger values of k and m cause Moonshine to analyze the input datastream in longer blocks, reducing the similarity of nearby patterns. The type of environmental source (audio, voltage, etc) also can change the relationship between parameter value and key quality.

In our implementation of Moonshine, an authenticator analyzes the noise source in real time to find the parameter values that maximize key quality and shares those parameters with the other device that is trying to pair. Both devices then use the shared parameter values to apply Algorithm 1, generating a shared key.

Moonshine uses a warmup period during which the device continuously samples the environmental noise source, building a bit sequence histogram. Once one device has collected enough data to characterize the properties of the environmental noise encoded by the histogram key generation commences. We use the histogram generated during warmup to define mappings from input bit sequences to output bit sequences during key generation.

Algorithm 1: Algorithm for Moonshine.

```
Input: A bit sequence B = {b_1, b_2, ..., b_N}
Input: Number of bits in a subsequence k
Input: Number of bits to skip between subsequences m
Output: A mapping from B → Z_{N-2}
for j ← 1 to k do
    sequences[j].frequency ← 0    // Initialization
    sequences[j].value ← 0
    sequences[j].order ← NULL
/* Count the frequency of occurrences of subsequences. */
for i ← 1 to N by m + k do
    subseq ← {b_i, b_{i+1}, ..., b_{i+k-1}}    // subseq is k-bit int
    sequences[subseq].frequency++
    sequences[subseq].value ← subseq
    if sequences[subseq].order, NULL then
        sequences[subseq].order ← i+k;
sequences ← sort(sequences)    // sort by frequency ascending
sequences ← sequences[1:N-2]    // Toss most-freq sequences
return sequences
```

5 EVALUATION OF MOONSHINE

In this section, we evaluate the performance of Moonshine with input data gathered from various forms of context-based authentication in real-world environments. Code is available from our GitHub repository[3]. Our ZIA datasets are available from [34].

[3] https://github.com/jweezy24/Moonshine

We show that Moonshine can generate high-quality keys from environmental noise sources with relatively low entropy rates, making them robust against attacks. Before presenting evaluation results, the following details the hardware and parameter settings as well as main evaluation metrics.

Datasets. The authors of VoltKey [16] lent us prototypes of their hardware, which allowed us to benchmark our prototype implementation of Moonshine on a realistic hardware platform. In addition to data gathered by VoltKey hardware, we used the ZIA datasets published by Fomichev et al. [10]. The dataset consists of data streams collected from seven types of sensors acceleration, luminosity, temperature, humidity, barometric pressure, magnetometer, gyroscope within different contexts. It includes long sequences of synchronized data from each sensor in an office setting, a mobile device, and a car. The authors preprocessed the data to generate bit sequences for each setting (Office, Mobile, Car). All preprocessed bit sequences from the dataset have low entropy density, caused mainly by long periods of inactivity in the underlying source noise processes. None of the bit sequences in the Office, Mobile, or Car datasets pass the NIST tests before being processed by Moonshine.

We also generated a dataset of radio frequency measurements similar to those published in [12, 19]. The hardware we used to collect the RF dataset consisted of a short stub of wire connected directly to the analog-to-digital converter input of a microcontroller. Our RF measurements produced relative high-quality bit sequences which passed most of the NIST tests before being processed by Moonshine.

The audio dataset [30] as well as the mobile, office, and car datasets [9] came from publicly available postings. To evaluate the audio, we created the bit stream using a commonly used algorithm [28]. Larger datasets are evaluated on a server-class machine because of memory restrictions on our Cortex M4. The server is also needed for NIST test evaluation, as the tests are not compatible with the Cortex M4 board. We also evaluate our algorithm on VoltKey hardware in real-time to characterize its performance on an IoT-class platform.

Evaluation metrics. We evaluate the performance of Moonshine with various metrics.

- NIST test pass rate: the fraction of NIST tests (discussed in §2.1) that pass. To generate high-quality keys, we want the NIST test pass rate to be as high as possible.
- Data retention rate: the fraction of input data retained in the output. We want to retain as much information as possible after processing by Moonshine.
- Diversity of datasets: We used a wide range of data from varying sources.

5.1 Subsequence Length

In this section, we want to understand how the key quality depends on the length of subsequences processed by Moonshine. According to the AEP (Ÿ2.2), longer subsequences should generate better keys. In Fig. 4(b) we plot the NIST test pass rate as a function of subsequence length for Moonshine-corrected data. We can achieve a near-perfect pass rate for subsequences of at least 6 bits. The same figure also plots the NIST test pass rate of the Von Neumann corrector for the same input data.

The tradeoff in Moonshine is that the corrector discards a portion of the input data stream in the process of distilling its randomness. Fig. 4(a) shows the amount of data remaining after randomness distillation.

Analyzing bits in longer subsequences increases the keys' quality because the typical set becomes more uniformly distributed on longer bit sequences. As the sequences that Moonshine considers become longer, there is a starker separation between the typical set and the non-typical set.[4] For long sequences (8 or more bits), the non-typical set represents a smaller fraction of the overall data, and more of the sequences that we actually observe are elements of the typical set. Since Moonshine retains elements of the typical set and discards elements of the non-typical set, it makes sense that we would retain more overall data if we consider longer bit sequences.

5.2 Run Time

In Fig. 4(c), we plot the run time of Moonshine on the VoltKey hardware, which uses an ARM Cortex M4 microcontroller running at 48 MHz. Moonshine does not add a substantial amount of time to the key generation process, even when processing long bit sequences. By comparison, the sampling, bit extraction, and key reconciliation steps take about 20 seconds on a pair of VoltKeys. This test ran on an input data stream of 200,000 bits gathered directly from the VoltKey. The distilled bit sequence is in the tens of thousands of bits in length, much longer than generally used for an authentication key. Moonshine can distill randomness from the input data sequence in a reasonable amount of time. The run time should not be substantially different when using different datasets as inputs.

5.3 Key Quality

In this section, we evaluate the quality of keys generated by Moonshine using the NIST test for randomness [25]. Moonshine makes two parameters available to the user: bit sequence length and bit drop length (k and m in Algorithm 1). Later evaluation will focus on the effect those parameters have on NIST test pass rates. Fig. 5 shows the NIST test pass rate as a function of k and m for the office, mobile, and car datasets, which generate raw data with similar entropy densities before being processed by Moonshine. Overall, Moonshine significantly improves the quality of keys generated from a wide variety of environmental noise sources.

Figure 3: Evaluation of Moonshine on VoltKey: (a) Percent of bits kept after Moonshine. (b) Percentage of NIST tests that pass. (c) Runtime of Moonshine on ARM Cortex-M4 microcontroller.

Choosing Parameters m and k. The general trend is that larger parameter values tend to generate higher-quality keys, so we would advise users of Moonshine to choose larger parameter values to improve key quality. This is because larger parameter values cause Moonshine to process the input data stream in longer blocks, resulting in less local similarity.

An exception to the trend is in the Car and Mobile 1 datasets, in which the most significant parameter values cause almost all NIST tests to fail. In those two datasets, the raw input datastream has such low randomness density already that Moonshine removes a substantial portion of the input bits when it is run with large parameter values. This creates problems because the NIST test suite needs a minimum number of bits to determine whether a bit sequence passes each test. The output data stream fails the NIST tests when Moonshine removes too many bits. We expect that keys generated from the Car and Mobile 1 datasets that fail with large parameter values would pass nearly all NIST tests if the datasets were larger. In practice, a system that used Moonshine to distill keys from low-entropy sources would gather more data from the environmental noise source to generate a high-quality key.

We found that the office, mobile, and car datasets have low randomness density and did not pass any NIST test before being processed by Moonshine because of their heavy reliance on human activity to generate randomness. During periods of inactivity such as a refuelling stop during data collection in the car result in long runs of ones or zeros in the generated bit sequences, causing the NIST tests to fail. Moonshine performs well even on low-quality datasets like Car and Mobile.

[4] This is the trend that is visualized in Fig. 2.

Figure 4: Histograms of random numbers generated by python's RNG compared to those generated by Moonshine and VoltKey . (a) 7-bit sequences after removing elements of nontypical set. (b) Raw 7-bit sequences taken from VoltKey. (c) 7-bit sequences from Python's RNG.

Fig. 4(b) shows the NIST test pass rate of Moonshine running on VoltKey as a function of bit sequence length. The entropy rate of VoltKey (discussed in Ÿ3.3) is much higher than the corresponding entropy rate of the car, office, and mobile datasets. The higher entropy rate of VoltKey gives us more entropy per key bit and better NIST test pass rates for the raw bit stream. Raw bit sequences generated by VoltKey pass just over half the tests in the NIST suite. Moonshine increases the randomness across all data sets we evaluated.

5.4 Discard Length

In this section, we test how the NIST test pass rate depends on the number of bits we discard (parameterized by m in Algorithm 1) when processing the input stream. We discussed the technique of discarding bits in the input stream in Ÿ4.1. Fig. 5 shows a heat map of the NIST test pass rate as a function of discard length (y-axis) and input sequence length (x-axis). Blue colors represent low NIST pass rates, and greens represent higher pass rates (lighter blues and greens are better).

In Fig. 5, there is a general correlation between data retention and NIST Test pass rate the more data Moonshine removes, the more NIST tests it passes, resulting in higher-quality keys. Furthermore, the datasets that are higher quality to begin with (such as, VoltKey , Audio, and RF) retain more data after applying Moonshine. The lower-quality datasets have more repetition that Moonshine must remove to produce a high-quality key.

5.5 Data Retention

In this section, we evaluate the fraction of input bits that remain in a key after being processed by Moonshine. Fig. 6 is a color graph which represents the percentage of data retained after Moonshine distills entropy from the input bit stream. Data retained is a function of the bit sequence length k and the discard length m.

VoltKey , which generates raw bit sequences with relatively high entropy rates, shows an increasing trend of data retained as bit sequence length increases. According to the AEP, longer sequences of samples will be more uniformly distributed and have a wider separation between the typical set and the nontypical set. Since Moonshine throws away elements of the nontypical set, there will be less information to throw away as sequence lengths get longer.

The office, mobile, and car data exhibit the same trend. However, the datasets retain less data. This is caused by the long periods of inactivity in those datasets. Long runs of ones and zeros on the input bit sequence represent a large percentage of the total bits extracted. Moonshine eliminates those long runs of ones and zeros even when m is small.

Moonshine works by selectively dropping bits from the raw input bit stream, so we expect that the output bit sequences will be shorter than the input. The bits that are dropped by Moonshine are those that carry the least amount of information.

6 DISCUSSION

In this section, we give some practical advice for designers of context-based authentication mechanisms which is based on the findings in this paper. We gave two methods of estimating the entropy rate encoded in an environmental noise signal: we calculate the approximate Rényi entropy rate of a Gaussian noise process in Theorem 3.4, and we give the more general form of a Shannon entropy rate of a noise process that includes a periodic signal plus additive white Gaussian noise in Equation 8.

Advice #1: Filter out as much periodic noise as possible from the environmental noise signal before digitizing. The goal of any context-based key generation scheme is to maximize the mutual information rate between two authenticators. Since mutual information of two random variables is upper-bounded by each random variable's entropy, we must maximize entropy of the individual source noise processes is to maximize mutual information.

Advice #2: Apply a Randomness Corrector to the Authentication Key Before Reconciliation. The raw bit sequences generated from environmental noise tend to have some predictable structure that causes them to fail the NIST test suite (Table 1). This seems to be caused by a low information density, or randomness per bit, in the extracted bit stream. Using Moonshine, we can distill the randomness from a long bit sequence into a shorter bit sequence, yielding a more secure key. The penalty that we pay when using a randomness corrector is that we must collect $2 \quad 3$ as many bits from the environmental noise process.

Figure 5: Fraction of NIST tests passed after applying Moonshine as a function of discard length (on the y-axis) and bit sequence length (on the x-axis).

Figure 6: Fraction of data retained after applying Moonshine as a function of discard length (on the y-axis) and bit sequence length (on the x-axis).

Advice #3: Use a High-Entropy Environmental Noise Source if it is Available. Moonshine is able to distill entropy from a high-entropy source of randomness like VaultKey to generate cryptographic keys that pass the NIST test suite. We expect that other context-based authentication systems that generate bit sequences that can pass around half of the NIST tests would bene t signi cantly from Moonshine. Moonshine is also able to improve key quality of bit sequences generated by extremely poor entropy sources like the mobile, o ce, and car datasets we analyzed in Ÿ5.

7 CONCLUSION

In this work, we presented Moonshine, a technique to distill randomness from low-entropy sources. We observed that keys generated from environmental noise sources are predictable largely because the distribution that of the random variable that we use to generate keys is often nonuniformly distributed. Our methods take advantage of the asymptotic equipartition theorem, which says that long sequences of iid samples from of any random variable will be almost uniformly distributed even if the the distribution of the individual samples is not uniform. Moonshine uses the AEP to identify and remove sequences of samples that are not uniformly distributed.

In our evaluation, we use the NIST test for randomness to evaluate the quality of keys generated by Moonshine operating on input data from several publicly available datasets. Our evaluation showed that Moonshine produces keys with substantially higher NIST test pass rates than the raw bit sequences extracted from multiple di erent environmental sources.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under award CNS-1845469, CNS-2003129, CNS-1838733, CNS-1719336, CNS-1647152, and CNS-1629833.

REFERENCES

[1] Taha Belkhouja, Xiaojiang Du, Amr Mohamed, Abdulla K. Al-Ali, and Mohsen Guizani. 2019. Biometric-based authentication scheme for Implantable Medical Devices during emergency situations. Future Generation Computer Systems 98 (2019), 109 119. https://doi.org/10.1016/j.future.2019.02.002

[2] C. H. Bennett, G. Brassard, C. Crepeau, and U. M. Maurer. 1995. Generalized Privacy Ampli cation. IEEE Trans. Inf. Theor. 41, 6 (Nov. 1995), 1915 1923. https://doi.org/10.1109/18.476316

[3] Christian Cachin and Ueli M. Maurer. 1997. Linking Information Reconciliation and Privacy Ampli cation. J. Cryptol. 10, 2 (March 1997), 97 110. https://doi.org/10.1007/s001459900023

[4] Thomas M. Cover and Joy A. Thomas. [n. d.]. Elements of Information Theory Wiley, Hoboken, New Jersey.

[5] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. 2004. Fuzzy extractors and cryptography, or how to use your ngerprints. In Proc. Eurocrypt, Vol. 4.

[6] Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakoubov. 2017. Fuzzy Password Authenticated Key Exchange. Cryptology ePrint Archive, Report 2017/1111. https://eprint.iacr.org/2017/1111.

[7] John E. Hershey, Amer Hassan, and Rao Yarlagadda. 1995. Unconventional cryptographic keying variable management. Communications, IEEE Transactions on 43 (02 1995), 3 6. https://doi.org/10.1109/26.385951

[8] Charles Eckert, Fatemeh Tehranipoor, and John A. Chandy. 2017. DRNG: DRAM-based random number generation using its startup value behavior. 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS) (2017), 1260 1263.

[9] Mikhail Fomichev, Max Maass, Lars Almon, Alejandro Molina, and Matthias Hollick. 2019. Index of supplementary les from "Perils of Zero- Interaction Security in the Internet of Things". https://doi.org/10.5281/zenodo.2537721

[10] Mikhail Fomichev, Max Maass, Lars Almon, Alejandro Molina, and Matthias Hollick. 2019. Processed data from Office scenario from "Perils of Zero-Interaction Security in the Internet of Things". https://doi.org/10.5281/zenodo.2537707

[11] John A. Gubner. 2006. *Probability and Random Processes for Electrical and Computer Engineers*. Cambridge University Press, New York, NY, USA.

[12] Suman Jana, Sriram Nandha Premnath, Mike Clark, Sneha K. Kasera, Neal Patwari, and Srikanth V. Krishnamurthy. 2009. On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environments. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom '09)*. ACM, New York, NY, USA, 321–332. https://doi.org/10.1145/1614320.1614356

[13] Ari Juels and Madhu Sudan. 2006. A fuzzy vault scheme. *Designs, Codes and Cryptography* 38, 2 (2006), 237–257.

[14] Ari Juels and Martin Wattenberg. 1999. A Fuzzy Commitment Scheme. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS '99)*. Association for Computing Machinery, New York, NY, USA, 28–36. https://doi.org/10.1145/319709.319714

[15] Andre Kalamandeen, Adin Scannell, Eyal de Lara, Anmol Sheth, and Anthony LaMarca. 2010. Ensemble: Cooperative Proximity-based Authentication. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*. ACM, New York, NY, USA, 331–344. https://doi.org/10.1145/1814433.1814466

[16] Kyuin Lee, Neil Klingensmith, Suman Banerjee, and Younghyun Kim. 2019. VoltKey: Continuous Secret Key Generation Based on Power Line Noise for Zero-Involvement Pairing and Authentication. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3, Article 93 (Sept. 2019), 26 pages. https://doi.org/10.1145/3351251

[17] Kyuin Lee, Neil Klingensmith, Dong He, Suman Banerjee, and Younghyun Kim. 2020. ivPair: Context-Based Fast Intra-Vehicle Device Pairing for Secure Wireless Connectivity. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20)*. Association for Computing Machinery, New York, NY, USA, 25–30. https://doi.org/10.1145/3395351.3399436

[18] Qi Lin, Weitao Xu, Jun Liu, Abdelwahed Khamis, Wen Hu, Mahbub Hassan, and Aruna Seneviratne. 2019. H2B: Heartbeat-based Secret Key Generation Using Piezo Vibration Sensors. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks (IPSN '19)*. ACM, New York, NY, USA, 265–276. https://doi.org/10.1145/3302506.3310406

[19] Suhas Mathur, Robert Miller, Alexander Varshavsky, Wade Trappe, and Narayan Mandayam. 2011. ProxiMate: Proximity-based Secure Pairing Using Ambient Wireless Signals. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys '11)*. ACM, New York, NY, USA, 211–224. https://doi.org/10.1145/1999995.2000016

[20] U. M. Maurer. 1993. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory* 39, 3 (1993), 733–742. https://doi.org/10.1109/18.256484

[21] Rene Mayrhofer and Hans Gellersen. 2007. Shake Well Before Use: Authentication Based on Accelerometer Data. In *Pervasive Computing*, Anthony LaMarca, Marc Langheinrich, and Khai N. Truong (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 144–161.

[22] Markus Miettinen, N. Asokan, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and Majid Sobhani. 2014. Context-Based Zero-Interaction Pairing and Key Evolution for Advanced Personal Devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, New York, NY, USA, 880–891. https://doi.org/10.1145/2660267.2660334

[23] Markus Miettinen, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and N. Asokan. 2018. Revisiting Context-based Authentication in IoT. In *Proceedings of the 55th Annual Design Automation Conference (DAC '18)*. ACM, New York, NY, USA, Article 32, 6 pages. https://doi.org/10.1145/3195970.3196106

[24] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. 2013. Heart-to-heart (H2H): authentication for implanted medical devices. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13)*. ACM, New York, NY, USA, 1099–1112. https://doi.org/10.1145/2508859.2516658

[25] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. [n. d.]. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. ([n. d.]).

[26] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. 2006. Secure device pairing based on a visual channel. In *2006 IEEE Symposium on Security and Privacy (S P'06)*. 6 pp.–313. https://doi.org/10.1109/SP.2006.35

[27] Dominik Schürmann and Stephan Sigg. 2013. Secure Communication Based on Ambient Audio. *IEEE Transactions on Mobile Computing* 12, 2 (Feb 2013), 358–370. https://doi.org/10.1109/TMC.2011.271

[28] D. Schürmann and S. Sigg. 2013. Secure Communication Based on Ambient Audio. *IEEE Transactions on Mobile Computing* 12, 2 (2013), 358–370. https://doi.org/10.1109/TMC.2011.271

[29] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N. Asokan. 2014. Drone to the Rescue: Relay-Resilient Authentication using Ambient Multi-sensing. In *Financial Cryptography and Data Security*, Nicolas Christin and Reihaneh Safavi-Naini (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 349–364.

[30] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. 2013. DEMAND: a collection of multi-channel recordings of acoustic noise in diverse environments. https://doi.org/10.5281/zenodo.1227121 Supported by Inria under the Associate Team Program VERSAMUS.

[31] Vincent van der Leest, Geert-Jan Schrijen, Helena Handschuh, and Pim Tuyls. 2010. Hardware Intrinsic Security from D Flip-Flops. In *Proceedings of the Fifth ACM Workshop on Scalable Trusted Computing (STC '10)*. Association for Computing Machinery, New York, NY, USA, 53–62. https://doi.org/10.1145/1867635.1867644

[32] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal de Lara. 2007. Amigo: Proximity-Based Authentication of Mobile Devices. In *UbiComp 2007: Ubiquitous Computing*, John Krumm, Gregory D. Abowd, Aruna Seneviratne, and Thomas Strang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 253–270.

[33] John Von Neumann. 1951. 13. various techniques used in connection with random digits. *Appl. Math Ser* 12, 36-38 (1951), 5.

[34] Jack West, Kyuin Lee, Younghyun Kim, George K. Thiruvathukal, and Neil Klingensmith. 2021. Moonshine Context-Based Authentication Harvested Data. https://doi.org/10.5281/zenodo.4579965

[35] Wei Xi, Chen Qian, Jinsong Han, Kun Zhao, Sheng Zhong, Xiang-Yang Li, and Jizhong Zhao. 2016. Instant and Robust Authentication and Key Agreement among Mobile Devices. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 616–627. https://doi.org/10.1145/2976749.2978298

[36] Lin Yang, Wei Wang, and Qian Zhang. 2016. Secret from Muscle: Enabling Secure Pairing with Electromyography. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM (SenSys '16)*. ACM, New York, NY, USA, 28–41. https://doi.org/10.1145/2994551.2994556

[37] Zhaoyang Zhang, Honggang Wang, Athanasios V. Vasilakos, and Hua Fang. 2012. ECG-Cryptography and Authentication in Body Area Networks. *IEEE Transactions on Information Technology in Biomedicine* 16, 6 (Nov 2012), 1070–1078. https://doi.org/10.1109/TITB.2012.2206115

Appendices

A PROOF OF THEOREM 3.1

Starting with the simple case of $N = 2$:

$$E\ D_{t_1} D_{t_2}\ = E\text{»}^1a_1 \cos^1 2\pi f t_1 + \Theta^o + a_2 \cos^1 4\pi f t_1 + \Theta^{oo} \\ {}^1a_1 \cos^1 2\pi f t_2 + \Theta^o + a_2 \cos^1 4\pi f t_2 + \Theta^{oo}\text{¼} \tag{9}$$

$$= E\ \ a_1^2 \cos^1 2\pi f t_1 + \Theta^o \cos^1 2\pi f t_2 + \Theta^o\ \ + \\ E\text{»}a_1 a_2 \cos^1 2\pi f t_1 + \Theta^o \cos^1 4\pi f t_2 + \Theta^o\text{¼} + \\ E\text{»}a_1 a_2 \cos^1 4\pi f t_1 + \Theta^o \cos^1 2\pi f t_2 + \Theta^o\text{¼} + \\ E\ \ a_2^2 \cos^1 4\pi f t_1 + \Theta^o \cos^1 4\pi f t_2 + \Theta^o \tag{10}$$

$$= \frac{a_1^2}{2} E\text{»}\cos^1 2\pi f\ ^1t_1 \quad t_2{}^{oo}\text{¼} + \\ \frac{a_2^2}{2} E\text{»}\cos^1 4\pi f\ ^1t_1 \quad t_2{}^{oo}\text{¼} \tag{11}$$

The cross terms in Eq. 10 evaluate to zero because the expectation is the inner product of two orthogonal sinusoids. The terms involving Θ in Eq. 10 evaluate to zero because we are taking the expectation over Θ, which is assumed to be uniformly distributed on $\text{»}\ \pi; \pi\text{¼}$. This leaves us with an autocorrelation function in Eq. 11 that depends only on the difference $^1t_1 \quad t_2{}^o$, which is the criterion for stationary. This same line of reasoning applies for $N > 2$. This result can also be ported to periodic signals by applying the appropriate limits. For $N > 2$, the same line of reasoning applies because expectations that do not involve Θ will be taken over triples of orthogonal sinusoids.

B PROOF OF THEOREM 3.3

Lemma B.1. *Let D and Z be random variables with alphabets D and Z respectively. Let $X = D + Z$ over alphabet X. Then:*

$$R^1X\,\mathrm{j}D^o = R^1Z\,\mathrm{j}D^o$$

Proof.

$$R^1X\,\mathrm{j}D^o = \tilde{\sum}_{d \in \mathsf{D}} P_D{}^1d^o R^1X\,\mathrm{j}D = d^o \tag{12}$$

$$= \tilde{\sum}_{d \in \mathsf{D}} P_D{}^1d^{o1} \ \log \tilde{\sum}_{x \in \mathsf{X}} P^1X = x\,\mathrm{j}D = d^{o2} \tag{13}$$

$$= \tilde{\sum}_{d \in \mathsf{D}} P_D{}^1d^{o1} \ \log \tilde{\sum}_{x \in \mathsf{X}} P^1Z = x \quad d\,\mathrm{j}D = d^{o2} \tag{14}$$

$$= \tilde{\sum}_{d \in \mathsf{D}} P_D{}^1d^{o1} \ \log \tilde{\sum}_{z \in \mathsf{Z}} P^1Z = z^0\mathrm{j}D = d^{o2} \tag{15}$$

$$= R^1Z\,\mathrm{j}D^o \tag{16}$$

After observing D, the remaining Rényi uncertainty in X is due to Z. Because conditioning reduces uncertainty:

$$R^1X^o \quad R^1X\,\mathrm{j}D^o = R^1Z\,\mathrm{j}D^o = R^1Z^o$$