Provably Secure Randomness Generation from Switching Probability of Magnetic Tunnel Junctions

Hong Jie Ng,[‡] Shuhan Yang,[‡] Zhaoyang Yao, Hyunsoo Yang,[*] and Charles Lim[†]
Department of Electrical & Computer Engineering, National University of Singapore, Singapore

(Received 15 October 2022; revised 15 February 2023; accepted 16 February 2023; published 23 March 2023)

In recent years, true random number generators (TRNGs) based on magnetic tunnel junctions (MTJs) have become increasingly attractive. This is because MTJ-based TRNGs offer some advantages over traditional CMOS-based TRNGs, such as smaller area and simpler structure. However, there has been no work thus far that has quantified the quality of the raw output of a MTJ-based TRNG and performed suitable randomness extraction to produce provably secure random bits, unlike the case for CMOS-based TRNGs. In the work presented here, we implement a MTJ-based TRNG and characterize the entropy of the raw output. Using this information, we perform postprocessing to extract a set of random bits that are provably secure.

DOI: 10.1103/PhysRevApplied.19.034077

I. INTRODUCTION

Random numbers are used in a myriad of applications, from Monte Carlo simulations for research to encryption for data security. Undoubtedly, random numbers are widely sought after, and much research has been done to construct random number generators (RNGs) that output high-quality random numbers. The quality of random numbers can be characterized by two properties—uniformity and predictability. In the ideal scenario, random numbers should be uniformly random and unpredictable from any perspective.

There are typically two classes of generators: *pseudorandom number generators* (PRNGs) and *true random number generators* (TRNGs). PRNGs make use of an algorithm, together with a short input random seed, to produce a longer sequence of random numbers that are uniformly distributed. However, PRNGs are deterministic algorithms at their core. Thus, any party with knowledge of the seed could predict the output of a PRNG perfectly. It is also possible for an observer to use the first few outputs of a PRNG to accurately predict the subsequent outputs. RandCrack [1], a Python script written in 2017, requires only the first 624 numbers from the Mersenne Twister [2] PRNG to be able to predict the following output numbers with high accuracy. In fact, the Mersenne Twister PRNG is widely used in C, MATLAB, and Python due to its supposedly strong generator properties.

Briefly speaking, TRNGs utilize randomness in physical processes (e.g., the time taken for atomic decay) to generate random numbers. These processes are inherently random and unpredictable. Ideally, TRNGs should (1) have high throughput, (2) consume low power, (3) be small in size, and (4) produce an output that is uniform and secret. Over the years, there have been many variations of TRNGs that have been implemented [3–9]. TRNGs based on magnetic tunnel junctions (MTJs) were proposed in 2014 [10]. In principle, a MTJ-based TRNG can be implemented using a topology of one transistor and one MTJ device. Therefore, compared with their CMOS-based counterparts, MTJ-based TRNGs can have a smaller area and a simpler structure. In addition, advanced MTJs can be switched by voltage pulses as short as a few nanoseconds, which makes MTJ-based TRNGs promising for performing high-speed and low-energy-consumption operations.

A MTJ device consists of a fixed layer and a free layer. Depending on the layers' relative magnetic alignment, the MTJ can exhibit two different stable states, namely a parallel (P) state and an antiparallel (AP) state. To switch between these states, we can take advantage of the spin transfer torque (STT), which allows us to utilize a current to manipulate the direction of the magnetic moment of the free layer [11]. In memory devices, current pulses are applied to set the MTJ into one of the stable states. However, by adjusting the amplitude and width of the current pulses, the MTJ can be perturbed into a metastable state, as shown in Fig. 1, where thermal fluctuations determine the final state of the MTJ and give rise to randomness [10]. It should be noted that

[*]eleyang@nus.edu.sg
[†]charles.lim@nus.edu.sg
[‡]These authors contributed equally to this work.

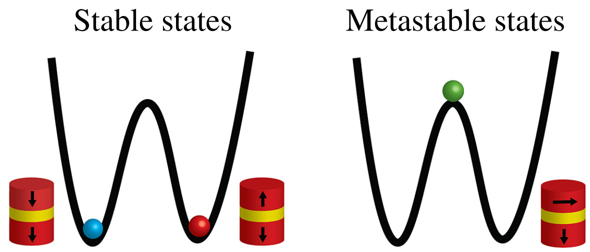

FIG. 1. Illustration of principle of random number generation with a STT-MTJ. Left, two stable low-energy states; right, metastable state.

the response of a STT-MTJ-based TRNG is sensitive to the width and amplitude of the input pulses. A fluctuation of the pulse width or amplitude can cause a deviation from a uniform distribution. Recently, TRNGs based on spin-orbit-torque (SOT) MTJs have been widely researched, and have been demonstrated to be promising for solving this issue [12]. Moreover, a TRNG based on a SOT-MTJ with perpendicular magnetization anisotropy has been proposed to have merits such as good power efficiency, fast switching speed, and better endurance [13]. However, the use of a SOT-MTJ-based TRNG for the generation of hundreds of megabits of random numbers, which is necessary for security characterization, has yet to be demonstrated.

A system is considered as *provably secure* if the system's security can be formally expressed or bounded using mathematical statements, under some general assumptions about the generator and its physics. This is in contrast to the usual heuristic analysis, which typically considers the bounded-adversary model, i.e., computationally bounded adversaries. A provably secure framework is very desirable, especially for important systems such as encryption systems, where their security has to be lasting and independent of technological advancements. In addition, such a classification provides an operational meaning as well. In the case of RNGs, this definition implies that an adversary will be unable to distinguish the output of the RNG from truly uniform and random data, except with a vanishingly small probability.

Thus far, previous implementations of MTJ-based TRNGs have not achieved provably secure random numbers. In one such study [14], entropy estimation was performed on a 1-Mbit dataset, which may be too small in size to eliminate statistical noise. As a result, the authors of that study did not perform suitable postprocessing to obtain provably secure random numbers. Other studies did not characterize the entropy of the raw output [10] or provided only a partial characterization of the raw output assuming that the raw bits were independent [15,16]. In the work presented here, we implement a MTJ-based TRNG setup and characterize the min-entropy of the raw output. Our min-entropy characterization is carried out with minimal assumptions, as we do not assume that the raw output bits are independent. We then perform suitable postprocessing in the form of randomness extraction to obtain a set of provably secure random numbers.

The remainder of this paper is organized as follows. In Sec. II, we provide implementation details of our work, including the experimental setup of our MTJ-based TRNG and the tuning of parameters. In Sec. III, we explain the idea behind provably secure randomness extraction and give a mathematical definition of the ideal output of a RNG. In Sec. IV, we characterize the raw output of our MTJ-based TRNG and choose a suitable function for performing randomness extraction. Lastly, we present the final postprocessed results in Sec. V and a conclusion in Sec. VI.

II. IMPLEMENTATION

In the present work, the TRNG is implemented using a STT-MTJ. Figure 2(a) shows the measured resistance-against-voltage (R-V) hysteresis curve of the fabricated MTJ device. The tunneling magnetoresistance ratio is approximately 200%. The writing operation of a STT-MTJ is a stochastic process, which is influenced by thermal fluctuations [17–19]. It is known that the switching probability of a MTJ device in the presence of thermal agitation can be expressed as [20]

$$P_{\text{sw}} = 1 - \exp\left\{-\frac{t}{\tau_0}\exp\left[-\Delta\left(1-\frac{I}{I_{c0}}\right)^2\right]\right\}, \qquad (1)$$

where t is the duration of the current pulse, τ_0 is the attempt time, Δ is the thermal-stability parameter of the nanomagnet, and I_{c0} is the critical switching current at 0 K. Figure 2(b) shows our experimental results for current-induced MTJ switching. The current-pulse width is 1 μs. We use Eq. (1) to fit our experimental data, and the data can be fitted well with the following parameters: $t = 1$ μs, $\tau_0 = 1$ ns, $\Delta = 104$, and $I_{c0} = 0.1032$ mA. Thus, in principle, we can obtain 50% switching by applying pulses 1 μs in width and 75.9 μA in amplitude. This makes it possible to implement a TRNG using a STT-MTJ. Moreover, because MTJ-based TRNGs can be integrated with CMOS chips at a high density [21], they could be a good candidate for future true random-number-generation devices.

Figure 3(a) shows the setup of our MTJ-based TRNG. The setup includes a pulse generator (Keysight 81134A) for generating the reset and perturb pulses, a data acquisition device (NI-USB 6363), and a 3-kΩ resistor for the reading operation. To guarantee simultaneous starting of the pulse-generation and reading operations, a trigger signal from the DAQ is applied to the pulse generator. The negative reset pulse from channel CH1 of the pulse generator and the positive perturb pulse from channel CH2 are combined in the power combiner. Because of the non-volatility of MTJ devices, a reset step is required before

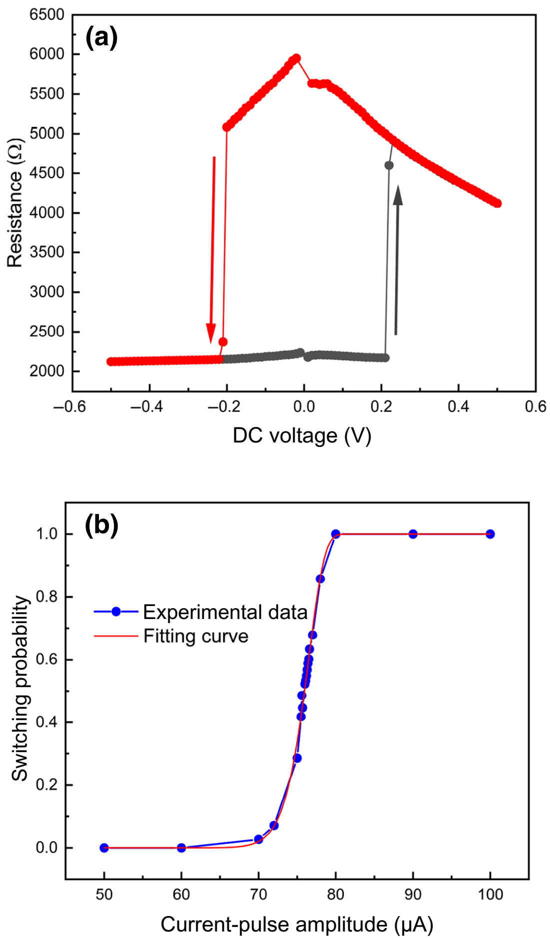

FIG. 2. Plots containing measurement results from our MTJ device. (a) *R-V* hysteresis curve. The gray (red) arrow represents P-to-AP (AP-to-P) switching. (b) Switching probability against current.

every bit of random number generation. Thus, a random-number-generation cycle consists of three steps: reset, perturb, and read [21].

For the purpose of reading, a dc offset voltage (V_{offset}) of 50 mV is superposed on the input signal. The resistance of the MTJ can be expressed as

$$R_{\text{MTJ}} = \frac{V_{\text{MTJ}}}{V_{\text{offset}} - V_{\text{MTJ}}} R, \tag{2}$$

where R_{MTJ} is the MTJ resistance, V_{MTJ} is the voltage drop across the MTJ, V_{offset} is the offset voltage applied for reading, and R is the resistance of the resistor in series with the MTJ. Figure 3(b) shows the time sequence for one random-bit-generation cycle. A negative reset pulse (V_{reset}) with an amplitude of -900 mV and a width of 1 μs is first applied to reset the MTJ to the low-resistance parallel state. After the reset pulse, a positive pulse (V_{perturb}) with an amplitude of 652 mV and a width of 1 μs is applied to perturb the MTJ to the metastable state. One cycle of random number generation takes 8 μs.

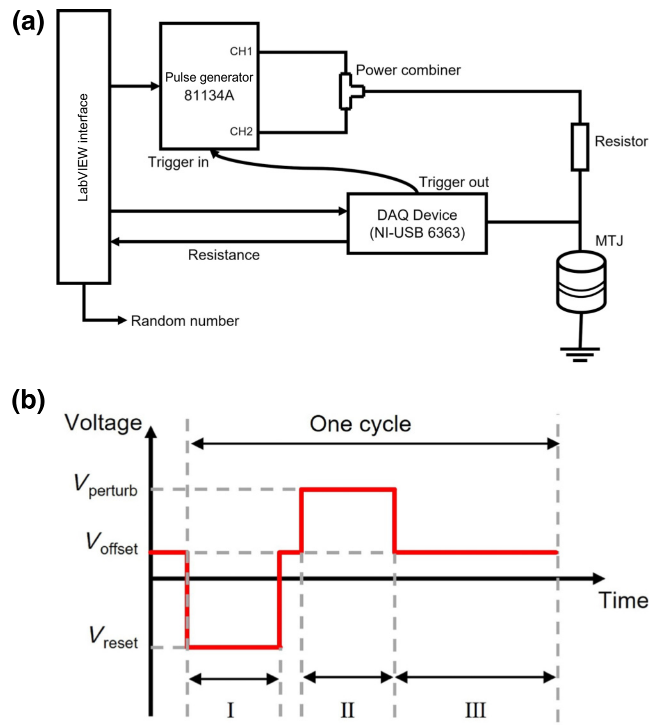

FIG. 3. (a) Random-number-generation setup. A Keysight 81134A pulse generator generates reset and perturb pulses. The trigger signal from the DAQ card ensures that the components are synchronized. The equipment is controlled by a LabVIEW interface. (b) Time sequence of random number generation. One cycle comprises three subsequences, which are I, reset operation; II, perturb operation; and III, reading operation.

The DAQ card performs the reading operation at 2 MHz. To obtain the MTJ resistance, it is required that the data points corresponding to the reset, perturb, and reading sequences can be clearly identified. In our case, a 1-μs pulse width is chosen to satisfy this requirement. After choosing the correct sequence of data points, we take the average and set 4 kΩ as a threshold in order to split the raw data into two bins. Finally, binary random numbers are output according to the final state that the MTJ is in.

In our implementation, the raw data are collected in batches of 250 000 bits, and the probabilities of 0s and 1s in each batch are computed. These correspond to the switching probability of the MTJ device. If the probability deviates by more than 0.1% from 50%, the input voltage applied to the MTJ device is tuned to shift the switching probability back to 50%. In real applications, this manual calibration process can be replaced by a feedback circuit that compensates for the deviation of the probability [12]. In addition, deviations in the MTJ response are accounted for during entropy estimation and, subsequently, randomness extraction, as we explain in the next section. Hence, our method is robust because we are still able to produce provably secure randomness even under nonideal conditions.

III. PROVABLY SECURE RANDOMNESS EXTRACTION FROM A WEAKLY RANDOM SOURCE

Consider a RNG that outputs a binary string. In the ideal scenario, the output has to satisfy two conditions, which are that (1) all the bits in the string are uniformly distributed, and (2) all the bits are independent from any perspective. We denote this ideal output by U_m. In our implementation (outlined in Sec. II), we focus on satisfying condition 1. We then perform rigorous entropy estimation (using techniques outlined in Sec. IV) to address condition 2.

However, despite our efforts, the raw output of our MTJ-based TRNG is most definitely not uniformly distributed. This is because the thermal noise is unlikely to remain constant throughout the entire experiment. Small deviations in the thermal noise skew the distribution of the output bits away from a uniform distribution. In addition, it would be naive to assume that the raw output bits are independent of each other. Imperfections in the experimental setup (e.g., a noisy reader) might introduce correlations into the raw output bits. In fact, the raw output data fail the permutation and chi-square independence tests outlined in the NIST 800-90B test suite [22].

As the n-bit raw output of our MTJ-based TRNG, which we denote by X, does not satisfy the two criteria stated above, it is classified as a weakly random source. We thus have to utilize functions known as randomness extractors (with the help of a seed, S) to extract an m-bit almost ideal output, Z, from the raw output of our MTJ-based TRNG. The process can be written as $Z = \mathrm{Ext}(X, S)$. We use the notion of statistical distance to determine how close Z is to the ideal output U_m.

The statistical distance between two random variables Y and Y' is defined as

$$\Delta(Y, Y') := \frac{1}{2}\sum_{y\in\mathscr{Y}} |P_Y(y) - P_{Y'}(y)|, \tag{3}$$

where $\mathscr{Y}$ is the range of values that Y and Y' can take. If Y and Y' are identically distributed, then their statistical distance is 0. On the other hand, if they are perfectly distinguishable, then their statistical distance is 1.

Intuitively, $m < n$, because it is not possible to generate randomness from nothing. This also means that for the extraction to be possible, X must contain at least m bits of randomness. To characterize the amount of extractable random bits that are present in a weakly random source, we use the notion of the min-entropy. The min-entropy of X is defined as

$$H_{\min}(X) := \min_{x\in\mathscr{X}}[-\log_2 P_X(x)]. \tag{4}$$

Notice that we use the min-entropy instead of the Shannon entropy to characterize the extractable randomness. To explain why, let us consider an extractor that outputs an ideal Z, i.e., $P_Z(z) = 2^{-m}$ for all $z \in \mathscr{Z}$. Any $z \in \mathscr{Z}$ thus has to satisfy

$$P_Z(z) = \sum_{x:\mathrm{Ext}(X)=z} P_X(x) = 2^{-m}. \tag{5}$$

Clearly, the implication is that $P_X(x) \le 2^{-m}$ for all $x \in \mathscr{X}$. Therefore, to extract m random bits from X, it is necessary that $H_{\min}(X) \ge m$.

Now that the preliminaries have been introduced, we provide a formal definition for randomness extraction.

Definition 1 (strong seeded extractors [23]).—The function $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$ is a (k, ε)-strong extractor if, for any source X with $H_{\min}(X) \ge k$ and a seed $S = U_r$, its output $Z = \mathrm{Ext}(X, S)$ satisfies $\Delta(ZU_r, U_mU_r) \le \varepsilon$, where YY' represents the concatenation of the strings Y and Y'.

The implication of a strong extractor is that the output Z remains uniformly random even in the presence of a seed S, i.e., Z and S are independent of each other. Operationally, this means that we are able to concatenate the seed to the output of the strong extractor, without any increase in the security parameter. This is important because in Toeplitz hashing, the seed used is longer than the output length. Thus, if we do not use a strong extractor, there will be no net randomness extracted, since we consume more randomness than we extract. We follow up by providing a definition of universal hashing.

Definition 2 (universal hash functions [24]).—A family of hash functions $\mathscr{F} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ with $n \ge m$ is called universal if, for f randomly chosen from $\mathscr{F}$,

$$\Pr[f(x) = f(x')] \le 2^{-m}, \quad \forall\, x \ne x' \in \{0,1\}^n. \tag{6}$$

In our application, we use a random seed S to choose a function from the family of universal hash functions. We denote the chosen function by f_s. This also means that the family of universal hash functions has size $|\mathscr{F}| = 2^r$. Importantly, Ref. [25] shows that universal hash functions can be used to construct strong seeded extractors. We provide a definition below.

Definition 3 (leftover hash lemma [25]).—Let X be a min-entropy source with $H_{\min}(X) \ge k$, and let $\mathscr{F} = \{f_s : \{0,1\}^n \to \{0,1\}^m\}$ be a universal hashing family of size 2^r with $m = \lfloor k - 2\log_2(1/\varepsilon) + 2 \rfloor$. Furthermore, let $\mathrm{Ext}(X, s) = f_s(X)$. Then, the extractor is a (k, ε)-strong extractor with seed length r and output length m.

Hence, using a family of universal hash functions to perform randomness extraction allows us to obtain an extracted output that is provably secure, i.e., we are able to bound how close our extracted output is to the ideal output. Another advantage is that such a security definition also adheres to the universal composability framework proposed in Refs. [26,27]. The implication is that if we use the extracted output in another protocol, the security parameter of that protocol will increase by ε.

IV. ENTROPY ESTIMATION AND EXTRACTION

As mentioned previously, our MTJ-TRNG produces an output that is weakly random, i.e., $H_{\min}(X) < n$. Thus, we have to lower-bound the min-entropy of X using techniques such as the NIST 800-90B entropy estimation suite [22] to certify that $H_{\min}(X) \geq k$. Using this estimate of k, we are able to quantify the amount of extractable randomness in X and extract an m-bit output string Z, such that $m \leq k$. The string produced is ε-close to the ideal uniform m-bit distribution U_m, i.e., $\Delta(Z, U_m) \leq \varepsilon$.

In the present work, we assume that one has access to a short uniformly random source, to be used as the seed. This might raise the question of why, if one already has access to a uniformly random source, does one still need to develop RNGs? This is because we have access to only a *short* seed, instead of having a source that can continuously output ideal uniform random numbers. As universal hashing is a strong extractor, the seed can be concatenated to the output without any additional loss in security. This results in randomness expansion, where the output randomness is greater than the input randomness.

The results that we obtain from NIST 800-90B indicate that the min-entropy per bit of X is 0.778 584. We emphasize again that we do not assume independence of the raw bits when performing min-entropy estimation. We collect $n = 217\,504\,350$ bits, which translates to $k = 169\,345\,406$ bits. We set our security parameter to $\varepsilon = 10^{-10}$, and employ the leftover hash lemma [25] to obtain $m = \lfloor k - 2\log_2(1/\varepsilon) + 2\rfloor = 169\,345\,260$ bits.

The extraction is carried out via Toeplitz hashing, which belongs to a family of universal hash functions [28]. Toeplitz hashing utilizes a Toeplitz matrix, $\mathbf{T}$. The Toeplitz matrix is an $m \times n$ diagonal-constant matrix and is constructed by filling the first column and first row of the matrix with S, the uniform seed. Thus, the seed length required for Toeplitz hashing is $n + m - 1$ bits. Z is obtained by performing $Z = \mathbf{T} \cdot X$, which is expressed as

$$\begin{bmatrix} s_n & s_{n-1} & \cdots & s_2 & s_1 \\ s_{n+1} & s_n & \cdots & s_3 & s_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{n+m-1} & s_{n+m-2} & \cdots & s_{m+1} & s_m \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}.$$

Toeplitz hashing has a computational complexity of $O(n^2)$, which is undesirable, as n needs to be a large number for us to accurately characterize the min-entropy of the raw data. To improve the computational complexity of Toeplitz hashing, we utilize the fast Fourier transform (FFT) algorithm. The FFT algorithm speeds up matrix-vector multiplications when the matrix used is a circulant matrix. The Toeplitz matrix $\mathbf{T}$ can be easily converted into a circulant matrix by padding $\mathbf{T}$ with additional rows and columns of 0s and 1s accordingly, to obtain $\mathbf{T}_{\text{circ}}$. Additionally, we pad X with 0s to get X_{pad}. The main process of Toeplitz hashing accelerated by the FFT can then be expressed as

$$Z_{\text{pad}} = \text{IFFT}(\text{FFT}(X_{\text{pad}}) \cdot \text{FFT}(T'_{\text{circ}})) = \mathbf{T}_{\text{circ}} \cdot X_{\text{pad}},$$

where FFT and IFFT are the FFT algorithm and its inverse, respectively, T'_{circ} is the first column of $\mathbf{T}_{\text{circ}}$, and Z_{pad} is the extended output. The original output Z can be recovered by keeping the first m bits of Z_{pad}. The computational complexity of Toeplitz hashing with the FFT is $O(n \log n)$, which is an exponential improvement.

V. RESULTS

We perform Toeplitz hashing on a personal computer with an Intel Xeon E5-2697 CPU and 128 GB of random access memory. The output from the Toeplitz hashing is saved in a text file, and the NIST 800-22 statistical test suite (STS) [29] is employed to test the quality of both the raw and the extracted random numbers. To run the test, the extracted (raw) data are split into 169 (217) blocks, each with length 10^6 bits. The blocks are passed to all 15 statistical tests, which perform hypothesis testing at the 1% significance level, and return two values, the P-value and the proportion.

The proportion parameter is the proportion of sequences that pass the test at the selected significance level. The range of acceptable proportions is determined using the confidence interval. It is calculated as

$$\widehat{p} \pm 3\sqrt{\frac{\widehat{p}(1-\widehat{p})}{k}},$$

where $\widehat{p} = 1 - \alpha$ and k is the number of sequences. When testing the extracted output, we substitute $\alpha = 0.01$ and $k = 169$ to obtain a confidence interval of 0.9670. The interpretation of this value is that if the returned value of the proportion for a specific test is below 0.9670, we conclude that our extracted output fails that test and is therefore not uniformly random. Similarly, when testing the raw output, we substitute $\alpha = 0.01$ and $k = 217$ to obtain a confidence interval of 0.9697.

TABLE I. Results of the NIST 800-22 statistical test suite for the raw and postprocessed data. The minimum values for each test are tabulated and rounded down to four decimal places.

	Raw data			Extracted data		
Statistical test	P-value	Proportion	Result	P-value	Proportion	Result
Frequency	0.0000	0.1901	Failure	0.2315	1.0000	Success
Block frequency	0.0000	0.8651	Failure	0.9233	0.9882	Success
Cumulative sums	0.0000	0.1839	Failure	0.4201	1.0000	Success
Runs				0.8606	0.9941	Success
Longest run	0.0000	0.1797	Failure	0.6436	1.0000	Success
Rank	0.1806	1.0000	Success	0.4547	0.9822	Success
FFT	0.0000	0.5253	Failure	0.4666	1.0000	Success
Nonoverlapping template	0.0000	0.0000	Failure	0.0001	0.9704	Success
Overlapping template	0.0000	0.0335	Failure	0.0024	0.9941	Success
Universal	0.0000	0.0000	Failure	0.6825	0.9882	Success
Approximate entropy				0.5659	0.9941	Success
Random excursions	0.0003	0.8182	Failure	0.2307	0.9703	Success
Random excursions variant	0.0909	0.9091	Success	0.0780	0.9604	Success
Serial	0.0000	0.0000	Failure	0.1597	0.9763	Success
Linear complexity	0.3781	0.9862	Success	0.3979	0.9882	Success

The P-value parameter returned for a specific test, denoted by P_T, is the result of the goodness-of-fit distributional test on the P-values obtained for that statistical test. The goodness-of-fit distributional test tests whether the distribution of P-values obtained follows a uniform distribution. If the returned P_T is less than 0.0001, then the P-values fail the uniformity test, and thus the sequence under test is considered to be nonuniformly distributed as well.

The results of the NIST tests for both our raw and our extracted data are displayed in Table I. For tests that are executed several times (e.g., the nonoverlapping-template

TABLE II. Comparison of features between existing TRNG implementations and our proposed MTJ-based TRNG. A check mark is placed in the "NIST 800-22" column if the implementation passes the NIST 800-22 STS. A check mark is placed in the "Provably secure" column if the study referenced performs suitable characterization of the min-entropy, and randomness extraction. Data that are not published is indicated with "..." in the table.

Reference	Entropy source	Throughput (Mb/s)	Efficiency (pJ/bit)	Area (μm^2)	NIST 800-22	Provably secure
[7]	CMOS metastability	162.5	9	957	✓	✓
[30]	CMOS jitter	9.9	42	920	✓	✗
[31]	CMOS metastability	86	6.08	10 000	✓	✗
[32]	CMOS metastability	1480	2.5	2 114	✓	✗
[9]	CMOS jitter	52	6.9	60 000	✓	✗
[10]	STT switching	66[a]	4.39[a]	$\sim$ 5.88[a]	✓	✗
[14]	STT switching	66	18	180	✓	✗
[16][b]	STT switching	101.5	6	404.8	✓	✗
[15][c]	STT switching	3.06	46.33	1.176	✓	✗
[12][d]	SOT switching	100	...	0.012	✓	✗
[13]	SOT switching	...	...	0.04	✓	✗
This work	STT switching	0.125	97	0.005[e]	✓	✓
This work (ideal)[f]	STT switching	100	0.832	0.005[e]	✓	✓

[a]These results are not given in the original reference, but are instead given in Ref. [15].
[b]This study was a theoretical one, i.e., the parameters were evaluated via a numerical simulation of eight MTJ devices operated in parallel.
[c]The parameters here are from using 24 MTJ devices in parallel.
[d]This study was a theoretical one, i.e., the parameters were evaluated via a numerical simulation of one SOT-MTJ device.
[e]This is the area of the MTJ pillar.
[f]The characterization of our MTJ shows that 50% switching can be realized by utilizing voltage pulses as short as 4 ns. The values in this row are calculated by assuming use of 4-ns pulses with an optimized electrical interface.

test), the minimum values from all the runs are tabulated. It can be seen that our raw data fail 10 out of 13 tests. The runs test is not performed on the raw data, because it carries out the frequency test as a prerequisite. As the frequency test returns a failure, the runs test is considered as failed by default. Similarly, because the serial test fails, the approximate entropy test is skipped as well.

On the other hand, after postprocessing, the *P*-values returned for all the tests are greater than 0.0001, and all the proportion values returned are also greater than 0.9670 as well. Hence, this shows that our extracted random numbers pass all tests in the NIST statistical test suite, and that we manage to extract high-quality random numbers from our initial weakly random source.

VI. DISCUSSION

In summary, we implement a MTJ-based TRNG, estimate the min-entropy of the raw output bits, and perform suitable postprocessing on the raw output to obtain a set of provably secure random numbers. The extracted random numbers pass all the tests in the NIST STS. Our implementation is also carried out with minimal assumptions, as we do not even assume that our raw data are independent or identically distributed when performing entropy estimation. We provide a comparison of our work with previous experimental TRNG implementations in Table II. As the present study is a proof-of-concept implementation, we do not aim to compete with other implementations in terms of power consumption or throughput. We are focused more on the security of our random numbers. As the speed of our implementation is highly limited by the data-transfer process, it could be greatly improved if combined with advanced CMOS technology.

Our work could have been further improved by performing an in-depth characterization of our setup (e.g., electrical noise in the DAQ) to model it using mathematical statements. By doing so, we would be able to obtain an accurate value of the min-entropy, instead of having to rely on min-entropy estimation tests.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ACKNOWLEDGMENTS

We thank Chao Wang and Jing Yan Haw for useful input and discussions. We acknowledge funding support from a National Research Foundation (NRF) of Singapore Fellowship grant (Grant No. NRFF11-2019-0001) and a NRF Quantum Engineering Programme 1.0 grant (Grant No. QEP-P2). This work is partially supported by the Advanced Research and Technology Innovation Centre (ARTIC), the National University of Singapore (Grant No. A-0005947-19-00), and the National Research Foundation Singapore (Grant No. NRF-000214-00).

[1] GitHub link: https://github.com/tna0y/Python-random-module-cracker.

[2] M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. **8**, 3 (1998).

[3] C. Petrie and J. Connelly, A noise-based IC random number generator for applications in cryptography, IEEE Trans. Circuits Syst. I: Fundam. Theory Appl. **47**, 615 (2000).

[4] R. Brederlow, R. Prakash, C. Paulus, and R. Thewes, in *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers* (IEEE, San Francisco, CA, USA, 2006), p. 1666, iSSN: 2376-8606.

[5] N. Liu, N. Pinckney, S. Hanson, D. Sylvester, and D. Blaauw, in *2011 Symposium on VLSI Circuits - Digest of Technical Papers* (IEEE, Kyoto, Japan, 2011), p. 216, iSSN: 2158-5636.

[6] Q. Tang, B. Kim, Y. Lao, K. K. Parhi, and C. H. Kim, in *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference* (IEEE, San Jose, CA, USA, 2014), p. 1, iSSN: 2152-3630.

[7] S. K. Mathew, D. Johnston, S. Satpathy, V. Suresh, P. Newman, M. A. Anders, H. Kaul, A. Agarwal, S. K. Hsu, G. Chen, and R. K. Krishnamurthy, μ RNG: A 300–950 mV, 323 Gbps/W all-digital full-entropy true random number generator in 14 nm FinFET CMOS, IEEE J. Solid-State Circuits **51**, 1695 (2016), conference Name: IEEE Journal of Solid-State Circuits.

[8] M. Kim, U. Ha, K. J. Lee, Y. Lee, and H.-J. Yoo, A 82-nW chaotic map true random number generator based on a sub-ranging SAR ADC, IEEE J. Solid-State Circuits **52**, 1953 (2017), conference Name: IEEE Journal of Solid-State Circuits.

[9] S. T. Chandrasekaran, V. E. G. Karnam, and A. Sanyal, 0.36-mW, 52-Mbps true random number generator based on a stochastic delta–sigma modulator, IEEE Solid-State Circuits Lett. **3**, 190 (2020).

[10] Won Ho Choi, Yang Lv Jongyeon Kim, A. Deshpande, Gyuseong Kang, Jian-Ping Wang, and C. H. Kim, in *2014 IEEE International Electron Devices Meeting* (IEEE, San Francisco, CA, USA, 2014), p. 12.5.1.

[11] A. Brataas, A. D. Kent, and H. Ohno, Current-induced torques in magnetic materials, Nat. Mater. **11**, 372 (2012).

[12] Y. Liu, Z. Wang, Z. Li, X. Wang, and W. Zhao, in *2018 IEEE 18th International Conference on Nanotechnology (IEEE-NANO)* (IEEE, Cork, Ireland, 2018), p. 1.

[13] H. Chen, S. Zhang, N. Xu, M. Song, X. Li, R. Li, Y. Zeng, J. Hong, and L. You, in *2018 IEEE International Electron Devices Meeting (IEDM)* (IEEE, San Francisco, CA, USA, 2018), p. 36.5.1.

[14] K. Yang, Q. Dong, Z. Wang, Y.-C. Shih, Y.-D. Chih, J. Chang, D. Blaauw, and D. Sylvester, in *2018 IEEE Symposium on VLSI Circuits* (IEEE, Honolulu, HI, 2018), p. 171.

[15] E. I. Vatajelu and G. Di Natale, High-entropy STT-MTJ-based TRNG, IEEE Trans. Very Large Scale Integration (VLSI) Syst. **27**, 491 (2019).

[16] B. Perach and s. kvatinsky, An asynchronous and low-power true random number generator using STT-MTJ, IEEE Trans. Very Large Scale Integration (VLSI) Syst. **27**, 2473 (2019).

[17] J. Slonczewski, Current-driven excitation of magnetic multilayers, J. Magn. Magn. Mater. **159**, L1 (1996).

[18] L. Berger, Emission of spin waves by a magnetic multilayer traversed by a current, Phys. Rev. B **54**, 9353 (1996).

[19] T. Seki, A. Fukushima, H. Kubota, K. Yakushiji, S. Yuasa, and K. Ando, Switching-probability distribution of spin-torque switching in MgO-based magnetic tunnel junctions, Appl. Phys. Lett. **99**, 112504 (2011).

[20] Y. Suzuki, A. A. Tulapurkar, Y. Shiota, and C. Chappert, in *Nanomagnetism and Spintronics* (Elsevier, 2014), p. 107.

[21] A. Fukushima, T. Seki, K. Yakushiji, H. Kubota, H. Imamura, S. Yuasa, and K. Ando, Spin dice: A scalable truly random number generator based on spintronics, Appl. Phys. Express **7**, 083001 (2014).

[22] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, *NIST 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation*, Tech. Rep. NIST SP 800-90b (National Institute of Standards and Technology, 2018).

[23] N. Nisan and D. Zuckerman, Randomness is linear in space, J. Comput. Syst. Sci. **52**, 43 (1996).

[24] J. Carter and M. N. Wegman, Universal classes of hash functions, J. Comput. Syst. Sci. **18**, 143 (1979).

[25] R. Impagliazzo, L. A. Levin, and M. Luby, in *Proceedings of the Twenty-First Annual ACM Symposium on Theory of computing* (Association for Computing Machinery, Seattle Washington USA, 1989), p. 12.

[26] R. Canetti, in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science* (IEEE, Newport Beach, CA, USA, 2001), p. 136, iSSN: 1552-5244.

[27] B. Pfitzmann and M. Waidner, in *Proceedings of the 7th ACM conference on Computer and communications security - CCS '00* (ACM Press, Athens, Greece, 2000), p. 245.

[28] H. Krawczyk, LFSR-based hashing and authentication, Adv. Cryptology-CRYPTO'94 **839**, 129 (1994).

[29] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, and J. Dray, *NIST 800-22 A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, Tech. Rep. (National Institute of Standards and Technology, 2010).

[30] E. Kim, M. Lee, and J.-J. Kim, in *2017 IEEE International Solid-State Circuits Conference (ISSCC)* (IEEE, San Francisco, CA, USA, 2017), p. 144.

[31] V. R. Pamula, X. Sun, S. Kim, F. u. Rahman, B. Zhang, and V. S. Sathe, in *2018 IEEE Symposium on VLSI Circuits* (IEEE, Honolulu, HI, USA, 2018), p. 1.

[32] S. K. Satpathy, S. K. Mathew, R. Kumar, V. Suresh, M. A. Anders, H. Kaul, A. Agarwal, S. Hsu, R. K. Krishnamurthy, and V. De, An all-digital unified physically unclonable function and true random number generator featuring self-calibrating hierarchical Von Neumann extraction in 14-nm tri-gate CMOS, IEEE J. Solid-State Circuits **54**, 1074 (2019).