frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 20 March 2023
DOI 10.3389/fcomp.2023.1157629

OPEN ACCESS

EDITED BY
Ruben Vazquez-Medina,
Instituto Politécnico Nacional (IPN), Mexico

REVIEWED BY
Aleksandra Mileva,
Goce Delcev University, North Macedonia
Jinjing Shi,
Central South University, China

*CORRESPONDENCE
Shantanu Chakrabartty
✉ shantanu@wustl.edu

SPECIALTY SECTION
This article was submitted to Computer Security, a section of the journal Frontiers in Computer Science

RECEIVED 02 February 2023
ACCEPTED 03 March 2023
PUBLISHED 20 March 2023

CITATION
Rahman M and Chakrabartty S (2023) GPS-free synchronized pseudo-random number generators for internet-of-things. *Front. Comput. Sci.* 5:1157629. doi: 10.3389/fcomp.2023.1157629

GPS-free synchronized pseudo-random number generators for internet-of-things

Mustafizur Rahman and Shantanu Chakrabartty*

Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO, United States

Introduction: Securing wireless communications in internet-of-things (IoT) requires both generation and synchronization of random numbers in real-time. However, resource constraints on an IoT device limit the use of computationally intensive random number generators and the use of global positioning systems (GPS) for synchronization. In this paper, we propose a synchronized pseudo-random number generator (SPRNG) that uses a combination of a fast, low-complexity linear-feedback-shift-register (LFSR) based PRNG and a slow but secure, synchronized seed generator based on self-powered timers.

Methods: A prototype synchronized self-powered timer (SSPT) array was fabricated in a standard silicon process and was used to generate dynamic random seeds for the LFSR. The SSPTs use quantum-mechanical tunneling of electrons to operate without any external power and are practically secure against tampering, snooping, and side-channel attacks (both power and electromagnetic).

Results: In this work, we explore protocols to periodically and securely generate random bits using the self-powered timers for seeding the LFSR. We also show that the time-varying random seeds extend and break the LFSR periodic cycles, thus making it difficult for an attacker to predict the random output or the random seed. Using the National Institute of Standards and Technology (NIST) test suite we verify the randomness of the measured seeds from the fabricated ensemble of SSPTs together with the random bit sequences generated by a software-seeded LFSR.

Discussions: In this modality, the proposed SPRNG could be used as a trusted platform module (TPM) on IoTs and used for verifying and authenticating secure transactions (e.g., software upgrades). Since the SPRNG system does not require access to GPS for synchronization, therefore it could be used in many resource-constrained and adversarial environments.

KEYWORDS

PRNG, self-powered, quantum-tunneling, LFSR, IoTs, synchronized-PRNG

1. Introduction

Random-number-generators (RNGs) play an important role in many applications ranging from optimization, game-theory, and simulations (Kroese and Rubinstein, 2012; Ma and Vandenbosch, 2012; Alimomeni et al., 2013). However, one of the most important uses of RNGs is in the area of secure communications (Schindler and Koç, 2009). Traditionally, this is achieved by encrypting the data using a sequence of random numbers i.e., cryptographic keys produced by an RNG. These keys are then synchronized using a timing reference extracted from a global-positioning-system (GPS) which also facilitates the exchange of encryption keys (Golino, 2014; Wilber, 2017). However, for battery-powered or passive internet-of-things (IoTs) devices where computational and energy resources are severely constrained this paradigm of secure communication using traditional RNGs is not practical. In this paper, we propose a novel RNG architecture that can be used for securing communications in IoTs.

RNGs fall into two major categories, namely, the true-RNGs (TRNG) and the pseudo-RNGs (PRNG). TRNGs generate random numbers based on non-deterministic physical processes such as thermal noise and entropy of natural phenomena (Sunar and Koç, 2009). Even though TRNGs are preferred for cryptographic applications, they are generally expensive and might not produce random numbers fast enough to be suitable for use in resource-constrained IoTs (Hsueh and Chen, 2019). On the other hand, a PRNG algorithm generates a sequence of numbers that is not truly random but whose statistical properties match that of a random number. In literature, there are many different types of PRNG that have been proposed (Bhattacharjee and Das, 2022). However, for resource-constrained IoTs the preferable PRNG is the one that is computationally inexpensive, fast, and can be easily fabricated and integrated into a System-on-Chip (SoC). In this regard, a Linear-Feedback-Shift-Register (LFSR) architecture is an optimal choice (Klein, 2013). It can be efficiently implemented using only flip-flops and XOR gates. An LFSR takes an initial value called seed as an input and generates each output bit with only a single shift operation, which satisfies both the low resource and the fast output requirement. However, one of the biggest challenges for an LFSR-based PRNG is the fact that the period is fixed and there is a need for reseeding to break the periodicity. While using a longer length LFSR or multiple LFSR would increase this periodicity, the problem still remains where once the period is reached the LFSR would start to repeat the random sequence. Furthermore, if the LFSR is seeded with a pre-stored static seed on boot-up, then it produces the same sequence of random numbers over and over again. One method to mitigate this issue would be to generate a dynamic seed. However, a resource-constrained IoT may not have access to a continuously running system clock or the GPS signal.

In addition to using a random number as a secure token, for secure communications there is also a need for synchronization of the tokens between the communicating parties. While asymmetric key encryption could be used to avoid this challenge, they are computationally too expensive to be universally implemented on these resource-constrained devices. On the other hand, a symmetric key encryption scheme can be customized for IoT platforms but requires a shared secret key (Henriques and Vernekar, 2017). Any static information stored on the IoT, such as a SecureID, used as the shared secret will be vulnerable to a machine learning type of attack (Maghrebi et al., 2016). Therefore, there is a need for a piece of dynamic information embedded into these IoT devices that can be synchronized in real-time. One such method for achieving this could be using a combination of a timing reference extracted from a global-positioning-system (GPS) and a timing reference generated locally using phased-locked oscillators. Unfortunately, in many IoT applications, this framework is impractical due to resource constraints together with the fact that many IoT devices may not have access to a GPS signal.

In this paper, we describe an architecture of a synchronized-PRNG (SPRNG) that can be used for generating synchronized pseudo-random binary sequences without the need for any GPS reference signal. The SPRNG uses a combination of a fast, low-complexity LFSR based PRNG and a slow but secure, synchronized seed generator based on our previously reported self-powered timers (Zhou and Chakrabartty, 2017; Mehta et al., 2022; Rahman et al., 2023). The self-powered timers use quantum-mechanical tunneling of electrons to operate without any external power and are practically secure against tampering, snooping, and side-channel attacks (both power and electromagnetic). In this work, we explore different protocols to periodically and securely generate synchronized random bits by seeding the LFSR using an array of self-powered timers. The concept is illustrated in Figure 1 in the context of IoT communications. The spatially separated IoT devices, IoT_A and IoT_B integrate a copy of the SPRNG for generating random tokens. The self-powered timers in these SPRNG form a clone where their temporal dynamics remain synchronized for long-period of time. When these synchronized-self-powered timers (SSPT) are used to dynamically seed the LFSR the random tokens generated by the LFSRs X and Y are precisely correlated. Therefore, these tokens can then be used as a shared secret key for facilitating secure communications between the IoTs. Furthermore, between power-ups, cold reboots, brown-outs, and system black-outs, the random keys generated using the approach shown in Figure 1 remain unique and aperiodic, which obviates the possibility of replay attacks.

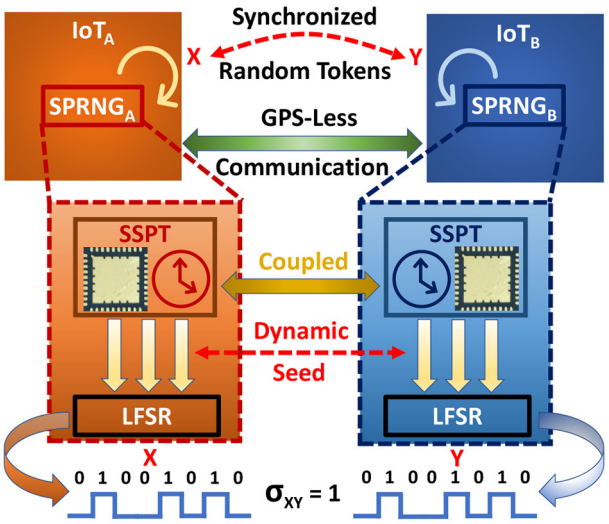

FIGURE 1
The concept of GPS-free secure Communication in spatially separated IoTs with SPRNGs: The IoTs generate random tokens using the SPRNG for use as cryptographic keys. The tokens are generated using a combination of a fast, low-complexity LFSR seeded by the Secure self-powered timers (SSPT). The synchronization of SSPT across both IoT_A and IoT_B ensures that the random tokens X and Y exhibit a perfect cross-correlation, $\sigma_{XY} = 1$.

2. Results

2.1. Secure self-powered timers and spatial synchronization

Figure 2A shows the micrograph of an array of self-powered timers along with the programming and readout circuit fabricated

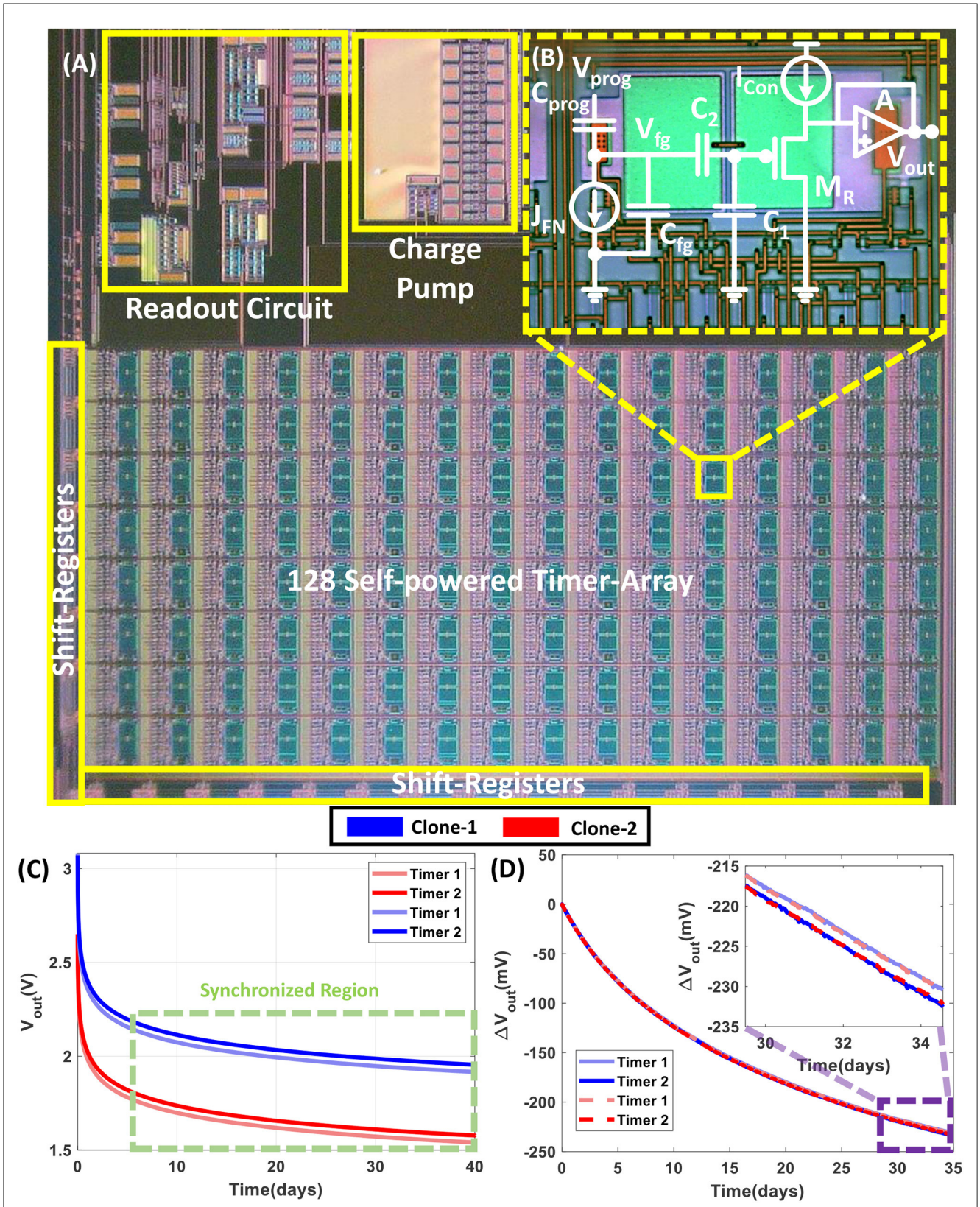

FIGURE 2
Measured dynamics and synchronization results using fabricated SSPT array: **(A)** Micrograph of an array of self-powered timers along with programming and readout circuit fabricated in a standard silicon process and **(B)** shows the micrograph of a single FN-timer in that array along with the equivalent circuit which is the building block of the SSPT. **(C)** The temporal response of the timers with different form factors on two different clones enters into the synchronized region after an initial settling stage. **(D)** Timers with the same form factors in the synchronized region have the same change in the output voltage over a long period of time across multiple clones.

in a standard silicon process. A simplified equivalent circuit model for each of the timers on the fabricated prototype is shown in Figure 2B. The operating principle of the timers involves injecting charge on an electrically isolated floating-gate capacitor C_{fg}. This is achieved by using a combination of hot-electron injection or quantum mechanical tunneling which are described in the Method Section 3.1. After the initial programming, the charge on C_{fg} is allowed to leak through the dielectric barrier, and is governed by the physics of Fowler-Nordheim (FN) quantum tunneling. Here the leakage current is denoted as J_{FN}. Note that the leakage process is thermodynamically and quantum-mechanically driven and hence does not require any external powering. This self-powered operation makes the timers immune to any power side-channel attack. Furthermore, J_{FN} is typically below attoamperes (or 10^{-18}A) which does not produce any measurable electromagnetic (EM) trace or fingerprint. Thus, the timers are practically immune to EM side-channel attacks. Furthermore, once the dynamics of the timers reach an *equilibrium* condition, any external probing using an EM source or using physical delamination disturbs the equilibrium and hence destroys the state of the timers. This implies that the self-powered timers are not only tamper-resistant but can only be copied through well-defined read-out mechanisms. Thus, we can assume that an array of FN-timers forms a secure dynamical system whose internal states could provide a secure mechanism for generating dynamic seeds for an LFSR.

In addition to its security features, FN-timers exhibit a unique synchronization feature where a pair of timers can be synchronized with each other, even if the devices are integrated on two different, spatially separated chipsets. A 'pair' of timers is defined as two timers designed with similar form factors. The synchronization feature is demonstrated by the experimental results in Figures 2C, D where we show the dynamics of two pairs of timers integrated on different chipsets that are spatially separated. Initially, the timers discharge quickly and the synchronization between different temporal dynamics is determined by device mismatch. However, as shown in Figure 2C, after a period of 5 days the temporal responses become "practically" independent of device mismatch and hence become synchronized to each other. This is shown in Figure 2D where after entering the *equilibrium* region, the dynamics of timer pairs remain synchronized. In our previous work (Zhou et al., 2019) we have shown that the timer pairs can maintain synchronization for a duration greater than a year. This implies that if we can derive the LFSR seed from the temporal response of the timers, then all IoT devices integrated with the SSPT can securely generate synchronized random numbers.

2.2. Secure seed exchange protocol

In order to use the synchronized random numbers as a cryptographic key for secure communication, the two IoT devices followed a simple protocol to synchronize their seeds. The seed generation protocol is described below:

1: IoT_A: Selects a set of 'N' timers to be sampled for generating the seed.
2: IoT_A: Measures the output of these timers to generate seed $S_A = \{V_{out}^i\}^N$, where $1 < i < N_T$ is the index of the timer, N_T is the total number of timer on the chip and V_{out} is the digitized output of the timers.
3: $IoT_A \rightarrow IoT_B$: Sends the indexes of the timer $\mathbf{I} = \{i\}^N$ along with the order of sampling.
4: IoT_B: Measures the output of all the timers in $\mathbf{I}$ in the specified order to generate $S_B = \{V_{out}^j\}^N$, where $j \in \mathbf{I}$ and V_{out} is the digitized output of the timers.
5: IoT_A, IoT_B: Generate random numbers based on the seed from the timers, $K_A = PRNG(S_A)$ and $K_B = PRNG(S_B)$. Here $PRNG()$ denotes the output of an LFSR seeded by S_A, S_B. K_A and K_B can then be used to encrypt and decrypt further communication.
Since the same set of timers $\mathbf{I}$, which are synchronized, is used for generating both S_A and S_B, therefore IoT_A and IoT_B have a common encryption key $K_E = K_A = K_B$.

SSPT Seed Exchange Protocol . Steps that IoT_A and IoT_B follows to obtain common encryption key K_E

IoT_A initiates the exchange protocol and generates a seed based on the digitized output of a set of timers. Information regarding the set of timers used by IoT_A is sent over an insecure public channel to IoT_B. On receiving this information IoT_B also generates a seed on its own. Once both seeds are generated the two IoTs can begin generating random numbers at higher-speed using an LFSR and start communicating using the synchronized random numbers as the encryption key.

Only Step 3 in the seed exchange protocol is assumed to be vulnerable as the communication is performed over a presumably insecure channel where an adversary can eavesdrop and learn this information. However, note that in order to derive the encryption key K_E the adversary needs to have access to one of the timer clones at the time of communication. However, by construction, only Iot_A and IoT_B have access to one of the clones and the adversary cannot clone or copy the timers (one of the security properties of the FN-timers). This means that the adversary cannot sample the hardware timers to generate a seed. In addition to this, we have also discussed in the previous Section 2.1 how the hardware timers are immune to any side-channel and snooping attacks. Thereby, it is also not possible for an adversary to deduce any information about the timers' output and generate the seed without actually sampling a clone. Note here that the actual output of the timers is also not accessible, only the random numbers from the LFSR are made attainable. This protects the seed exchange protocol from any kind of regression or Machine learning attack for predicting the output of the timers. In our previous work (Rahman et al., 2022) we show that the parameters determining the temporal response of the timers cannot be determined by an adversary without the knowledge of the initialization condition. Furthermore, since the seed is derived from a dynamic process, it will change with time

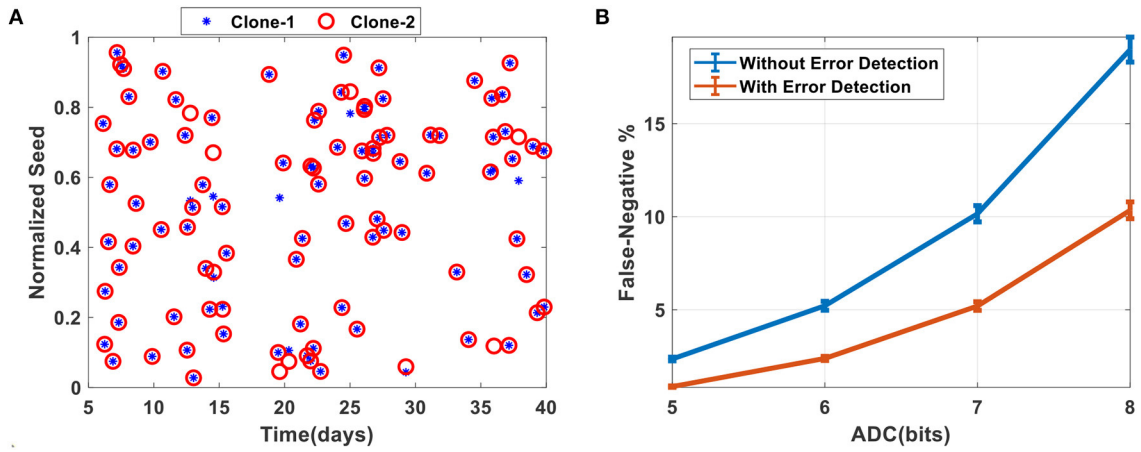

FIGURE 3
Random Seed Generation and Synchronization from fabricated prototypes: **(A)** The normalized seed generated from the SSPT at different times are spread evenly across the domain. **(B)** The percentage of False-Negative in the synchronization of two valid seeds due to readout and quantization error.

thereby breaking the period of the LFSR. In this regard, the output of the LFSR will appear to be a 'true' random number for any adversary.

2.3. Noise robustness of seed exchange protocol

In the next set of experiments, we quantified the noise robustness of the protocol using a fabricated FN-timer array. We generated seeds from two fabricated prototypes of SSPT using the same set of timers. The details of the experiment are provided in the Method Section 3.2. Figure 3A shows the normalized seeds generated from both clones sampled at different time instances. We can observe that the seeds derived from the temporal response of the timers are uniformly distributed across the whole dynamic range with time. This implies that the seeds are unpredictable without knowledge of the underlying principle, timers' output. However, we do observe that there are a few mismatches among the seeds from the two clones. This is due to the readout and quantization noise of the analog-to-digital converter (ADC). To mitigate this issue a lower-resolution ADC can be used. In order to find out the expected number of mismatches at different resolutions of ADC we performed a Monte Carlo study where we generated seeds at random time instances with 5, 6, 7, and 8 bits ADC (details in Method |Section 3.2). Figure 3B shows that as we decrease the resolution of the ADC, the percentage of mismatches between valid seeds i.e., False-Negatives, also decreases. However, this comes at a cost of the security of the protocol. This is because using lower-resolution ADC would result in less frequent changes in the value of the seed and might not be enough to break the period of the LFSR. Therefore, a tradeoff exists between the security and robustness of the protocol. Another method that could be used to reduce the possibility of False-negatives is by using error correcting code such as Cyclic-Redundancy-Check (CRC). Even with a simple CRC code of size 3 bits detecting at least 2 bits hamming distance between the two seeds the percentage of False-Negatives can be reduced at all resolutions of ADC as shown in Figure 3B. The details of this experiment are provided in the Method–Section 3.2. Note that, this improvement in accuracy comes at a cost of more computational resources required for the protocol. Thereby the usage of such methods would depend on the application and resource availability of the IoT device in question and the demand for accuracy.

2.4. Statistical test for SPRNG

In order to evaluate the randomness of the numbers generated by SPRNG we performed benchmark tests using the Statistical Randomness Test Suite (SP800-22 Rev 1a) made available by the National Institute of Standards and Technology (NIST) (Bassham et al., 2010). The suite consists of 15 statistical tests the results of which are represented in a form of *P*-values in the range [0, 1]. A binary string is tested to be random if the *P*-value exceeds a certain threshold value in all 15 tests. This threshold value was chosen to be 0.01, as recommended by the NIST specification, which suggests that the string is random with a probability of 99%. The details of the experiment are provided in the Method–Section 3.3 and the results of all 15 tests are tabulated in Table 1.

The first experiment was done with a single LFSR as the random number generator seeded with the digitized output of the timers. We observe that the minimum pass rate is approximately 91 for a sample of 100 binary strings, in the case of the Linear Complexity Test and Random Excursion Test. These results could be further improved by using two independent LFSRs of different sizes, randomly seeded by the SSPT, and then XORing the output of them to generate the random binary strings. In this case, the minimum pass rate is 96 out of a sample of 100 binary strings. Note here that this technique not only improves the quality of random numbers generated but also increases the periodicity of the overall sequences. This would ultimately increase the lifetime of the SSPT as discussed in the following section. However, this

TABLE 1 Results showing the randomness of the numbers generated by SPRNG when tested with NIST test suites for both cases, a single LFSR and when two LFSR are XORed.

NIST TESTs	Single LFSR		XORed LFSRs	
	Average *P*-value	Pass ratio	Average *P*-value	Pass ratio
Monobit test	0.46	100/100	0.556	100/100
Frequency within block test	0.4137	98/100	0.521	98/100
Runs test	0.4492	100/100	0.5427	100/100
Longest run ones in a block test	0.5499	100/100	0.3685	100/100
Binary matrix rank test	0.5407	100/100	0.4759	100/100
DFT test	0.5122	100/100	0.4938	100/100
Non-overlapping template matching test	1.000	100/100	1.000	100/100
Overlapping template matching test	0.5296	100/100	0.4872	100/100
Maurers universal test	0.4919	100/100	0.5518	100/100
Linear complexity test	0.2757	91/100	0.4736	100/100
Serial test	0.3459	97/100	0.4394	100/100
Approximate entropy test	0.4212	100/100	0.5553	100/100
Cumulative sums test	0.3898	100/100	0.4046	100/100
Random Excursion Test	0.0995	91/100	0.1581	96/100
Random excursion variant test	0.1318	93/100	0.1251	96/100

comes at a cost of efficiency of the SPRNG as more measurements and computation are needed to be done. Therefore, a tradeoff exists between efficiency and lifetime and security. Depending on the application (how long the IoT will be in use) and specification (how secure the communication needs to be) of the IoT device either single-LFSR or double-LFSR implementation of SPRNG can be used. Nevertheless, from the results in Table 1, we can definitely conclude that the bit strings generated by the SPRNG, both with single and double LFSRs implementation, are statistically random in nature.

2.5. SSPT lifetime analysis

In our previous work (Zhou and Chakrabartty, 2017; Zhou et al., 2019) we have shown that the temporal response of the fabricated timers can be modeled as

$$V_{out}(t) = \frac{k_2}{\log(k_1 t + k_0)} \tag{1}$$

Where k_1, k_2 are device specific and fabrication specific parameters, $k_0 = \exp\left(\frac{k_2}{V_0}\right)$, V_0 refers to the initial voltage at the floating-gate, and t refers to the time elapsed after initialization. The detailed derivation is excluded here for the sake of brevity and can be found in Zhou and Chakrabartty (2017) and Zhou et al. (2019). In Figure 4A we can observe that this analytical model can accurately track the output of the fabricated timers once the parameters are regressed from the measured data. The details of the regression process are provided in the Method Section 3.4. We use the analytical model to determine the lifetime of the SSPT. Note that since the timers are initialized with a fixed amount of charge, the dynamics of timers will slow down to single-electron tunneling events. The question being analyzed here is whether an ensemble of FN-timers can still exhibit state-change that is faster than the period of the LFSR. Figure 4B, i shows the state of the LSB of the digitized output for a single timer. Note here that in order for the dynamic seed to change only a single-bit flip of the digitized output would suffice. Therefore, the change in the LSB state represents the change in the dynamic seed. Since, the dynamical system slows down non-linearly as time passes the rate of change of the dynamic seed will also decrease over time, as evident from Figure 4B, i. However, to break the periodicity of the LFSR the dynamic seed needs to change before we generate the maximum length of a random number. For example, let an LFSR generate random numbers with a clock speed of 1 GHz. Then the LFSR with seed length of 49 bits will generate the maximum length of a random number in $\frac{2^{49}}{10^9}$s, which is $\approx$ 6.5 days. Now, if a single timer was used to generate the seed in this case, then from Figure 4B, i we can derive that after a period of $\approx$ 60 days, the dynamic seed no longer breaks the periodicity of the LFSR. We denote this period as the 'lifetime' of the SPRNG, as after this period the random sequence will start to repeat itself. The lifetime of the SPRNG can be increased by increasing the resolution of the ADC used for digitizing the output since this would change the seed more frequently. However, as shown previously, this would come at the cost of the seed exchange accuracy of the protocol. Another method to increase the lifetime would be to use multiple timers for generating the dynamic seed. This is because as the number of timers used for generating the seed increases, the probability that at least one of the timer's digitized output changes also increases. This can be observed from Figure 4B, ii where three timers were used to generate the dynamic seed which subsequently increases the lifetime of SPRNG. Furthermore, while using multiple timers, the order in which the digitized output of the timers is sequenced

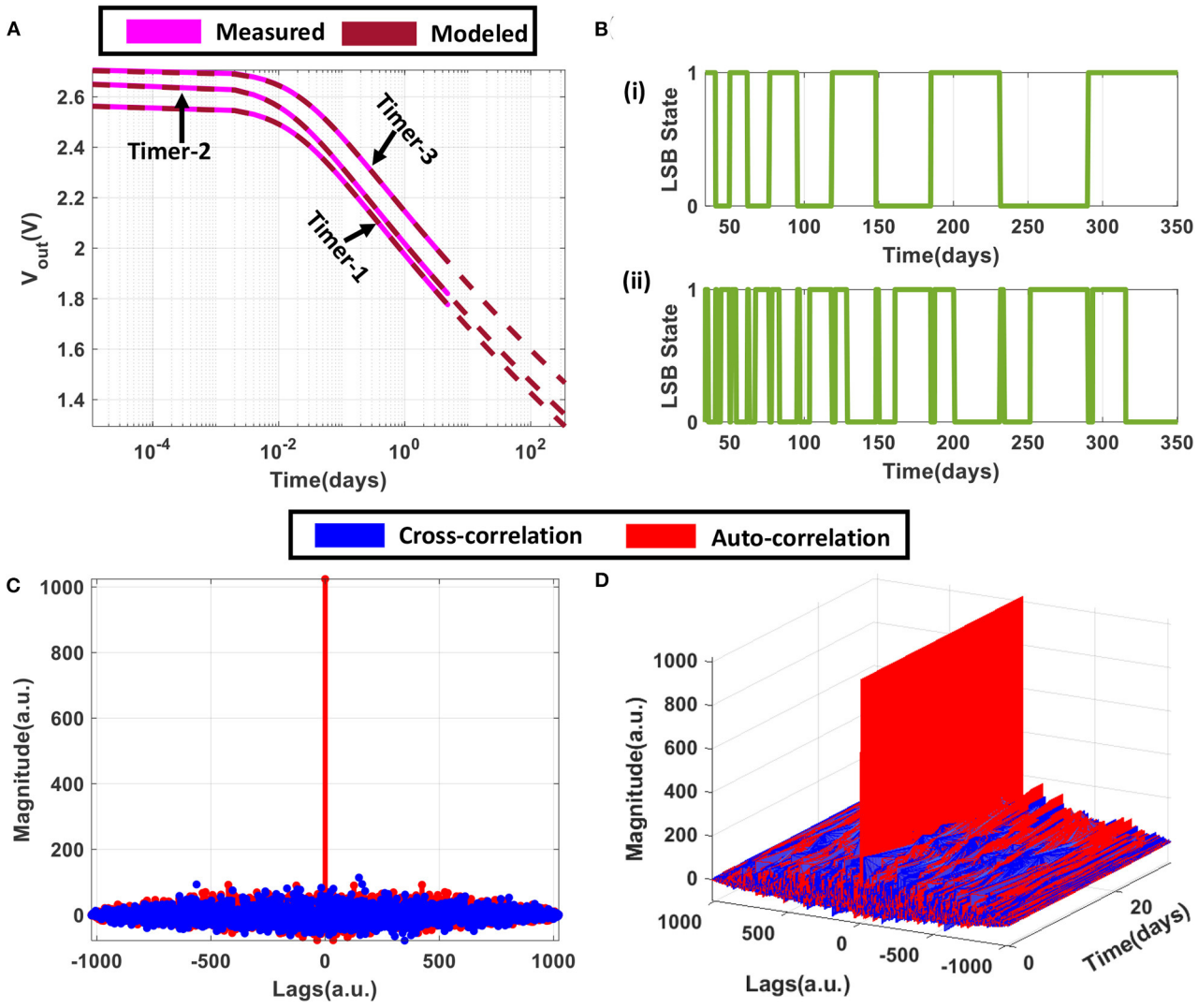

FIGURE 4
Correlation between the outputs of LFSR for shifted seed: **(A)** Accurate representation of the fabricated timers by its analytical modeled counterpart. **(B)** The state of the LSB for the digitized output of the fabricated (i) Timer-1 in **(A)** and (ii) Timer-1, 2, and 3 in **(A)** XORed. **(C)** Autocorrelation for the output of LFSR for a particular seed along with the cross-correlation with another output generated at the same time instance with the order of seed generation changed. **(D)** The results when the procedure in **(C)** is repeated across multiple different time instances.

can also be changed to break the periodicity of the LFSR. This can be observed in Figure 4C which shows the correlation between two random numbers generated with the same set of timers sampled at the same instance, but only the order of sequencing their digitized output to generate the seed is changed. In order to obtain a reference for the noise floor we also calculated the autocorrelation of one of the random numbers. Figure 4C shows that when the lag in the case of autocorrelation is zero, the correlation is at maximum. For any other lag, the autocorrelation is 5 times less than that of the maximum magnitude meaning that there is hardly any similarity or periodicity in the random number itself. This is trivial for a random number as there should not be any correlation between two blocks of sequences within the same number. Next, we observe that the magnitude of the cross-correlation between the two random numbers is also within this range. Therefore, we can conclude that changing the order of the timers' sequence while generating the dynamic seed will also break the periodicity of the LFSR. Figure 4D shows that this is true across all time instances.

3. Methods

3.1. Programming the SSPT

The programming of the SSPT requires injecting charges on the electrically isolated floating gates such that the floating-gate potential (V_{fg} in Figure 2B) can be set to a level at which FN-tunneling is measurable. This is accomplished by setting V_{prog} to a high-potential of 22V using an internal (on-chip) charge-pump. After the initial programming, the floating-gate is allowed to discharge while the potential V_{out} is measured periodically. The measurement is performed using a capacitive voltage divider

formed by C_1 and C_2 such that the attenuated voltage can be measured using standard readout buffers. Furthermore, since the tunneling nodes are electrically isolated we use a readout MOSFET M_r configured as a source-follower using a constant current-source I_r to read the voltage V_{fg} at C_1. The voltage of the source follower is buffered using A to avoid any coupling to the tunneling junction. The readout voltage was programmed to around 3V during the initialization of the tunneling node. For the results shown in Figures 2C, D, each individual timer cell, shown in Figure 2A, on two separate timer pairs having the same configuration was initialized independently to a high FN-tunneling region. After the one-time programming, V_{prog} was set to 0V and the timers discharged naturally. In this mode of operation, no external power is required. The readout voltages of each timer cell on both the clones were measured every 180s for a duration of over 40 days.

3.2. Seed generation

The measured outputs from the two hardware clones were used to generate the seed at each time instance. The analog readout voltages V_{out} for each timer cell were quantized using an ADC to generate a binary string. These binary strings of multiple timers were concatenated to generate seeds of variable length depending on the size of the LFSR used. For the results shown in Figure 3A 7 timers were used from each clone to generate the seed with their outputs quantized to 7 bits precision. This resulted in a seed of length 49 bits. The measured outputs were sampled randomly at 100 different time instances and the generated seeds were normalized for visual comparison.

For results shown in Figure 3B we generated 1,000 seeds at random time instances using the same procedure as discussed above where measured outputs were quantized with $5, 6, 7$, and 8 bits precision. Each instance of sampling where the seeds from both hardware clones did not match perfectly was counted as a False-negative. The experiment was repeated 1,000 times, each time the sampling and seed generation was performed on a different set of random time instances. This represented the case where no error detection was used. In the case of error detection, the digitized seeds are represented as the coefficients of a message polynomial which is then divided by a pre-determined generator polynomial to calculate the CRC bits i.e., the coefficient of the remainder polynomial. In the seed exchange protocol at Step-3 IoT_A sends the CRC bits along with the other information. IoT_B can use these CRC bits to check whether the seed that it generated, S_B, matches with that of IoT_A. For a generator polynomial of size 3-bits, IoT_B can detect at least 2-bits of error (Koopman, 2015). Therefore, in Figure 3B, seeds from two clones with a hamming distance of 2 or less were not counted as False-negative. The mean and variance of the percentage of False-negative were calculated across all experiments.

3.3. Randomness test

The seeds for the LFSR were generated using the measured output from the hardware clones as described in the previous sections. These seeds were then fed into an LFSR which was simulated in MATLAB. In the case of the Single LFSR, shown in Table 1, the length of the LFSR chosen was 49 bits, which means 7 timers' output was used each quantized with an ADC of 7 bit precision. The seeds were generated across 100 random time instances and corresponding to each seed 1 MiB ($2^{20} bits$) were simulated from the LFSR. These bit strings were then tested with the NIST SP800-22 Rev 1a PRNG test suite using the Python implementation by David Johnston (Johnston, 2021). This process was repeated for the XORed LFSR, however this time two sets of seeds were generated at each time instance. The length of one of the LFSRs used in this case was 49 bits, the same as before, and the other one was 42 bits. To generate the seeds 7 and 6 timers' outputs were used respectively with a 7 bit precision ADC. The individual outputs of the LFSRs, 1 MiB, were then XORed with each other for producing the random bits which was then tested in a similar manner as before.

3.4. Extending SSPT lifetime through shifted seed generation

The measured output shown in Figure 2C was used to regress the parameters k_0, k_1, and k_2 as shown in equation 1 for each timer in the fabricated prototype. Even though each timer's output was measured for a period of $\approx$ 40 days, only the data for the first 5 days were used to regress the parameter. In this manner, we could verify that the regressed parameters, when used to represent the measured results, accurately predicted the temporal response of each timer against the measured result for the rest of the 35 days. This is validated in Figure 4A.

Each of the timer cells in the fabricated prototype can be selected for reading out the output values using a serial shift-register. However, depending on the order of read-out of these timer cells the seed that is generated from the quantization of their output is different for every permutation. This means that with the same set of timers multiple seeds can be generated. For the results shown in Figure 4C, two sets of seeds were generated at the same time instance using the same set of 7 timers with only the order of the timers shifted by one in a cyclic manner. For example, if one of the seeds was generated using the order $[19, 45, 54, 61, 89, 119, 120]$, then the order for the other seed was $[120, 19, 45, 54, 61, 89, 119]$. These seeds were then used by the same LFSR (length 49 bits) and 1 KiB of random binary strings were generated. The auto-correlation of one of the binary strings was calculated along with the cross-correlation with the other binary strings. Note here that for these calculations the binary states were represented as $[-1, 1]$ instead of $[0, 1]$. This process was repeated across 1,000 random time instances, the result for which is shown in Figure 4D.

4. Discussion

In this paper, we described an architecture of a light-weight synchronized-pseudo-random-number generator (SPRNG) that can be used for securing wireless communications in IoTs. The solution does not require access to GPS and therefore could be used in many resource-constrained and adversarial environments. Some of the applications include personal IoTs used in health-care (Baker

et al., 2017), key-fobs (Eddy, 2022) to military-grade IoTs that need to operate in RF-jamming environments (Staniec and Kowal, 2020). The combination of ultra-secure slow-dynamics exhibited by the FN-timers and fast-dynamics exhibited by standard LFSR provides an ultra-fast and yet secure mechanism to generate secure tokens that could potentially be used for high-speed transactions (Yukonhiatou et al., 2020). However, note that for this application, clock-frequency and clock-phase synchronization between the communicating devices are required and have not been addressed in the paper. The inherent security of the proposed approach lies in the *no-cloning* property of the FN-timers, therefore, only the communicating IoTs will have access to the secure random tokens. During each communication session, and even after a cold reboot the tokens are randomly generated and hence an adversary cannot initiate a replay attack.

A potential limitation of SPRNG proposed in this work in cryptographic applications arises due to the usage of an LFSR as the PRNG. If an adversary manages to extract $2L$ bits of the LFSR output, where L is the length of the dynamic seed, then by using Berlekamp-Massey algorithm (Massey, 1969) they can represent the LFSR in an analytic form. This significantly reduces the lifetime of SPRNG since the LFSR are now needed to be seeded much more frequently. One method to achieve this would be to use a different set of timers to dynamically seed the LFSR every $2L$ bits. Another method to mitigate this issue would be to use an Alternating Step Generator (ASG) proposed by Günther (1988) where three LFSRs are used in conjunction to produce the random sequences. Note that in this implementation all three LFSRs would be dynamically seeded by three different sets of timers and the synchronization between the two random tokens on spatially separated devices can still be achieved. The best possible attack in this scenario that can be mounted will require $\mathcal{O}(2^{\frac{2L}{3}})$ bits (Khazaei et al., 2007). This practically ensures that the lifetime analysis in the Results–Section 2.5 remains valid.

One consideration that has not been discussed before in the paper is the effect of environmental variations (for example temperature) on the synchronization of the FN-timers. In Zhou et al. (2019) we reported that the dynamics of FN-timers exhibit a temperature dependence, however, when the temperature remains static, the dynamics of the FN-timer still remain synchronized with respect to each other. Therefore, one of the key requirements for the proposed SPRNG-based secure communication is to ensure proper temperature controls. However, this feature could also be used to further enhance security where the operating temperature could be treated as private information that is only known to the communicating parties.

Future work in this area would require developing a complete system-on-chip solution where the SPRNG acts as a core trusted-platform-module (TPM) that like the commercial AES core in secure processors can generate tokens for use by the rest of the SoC modules.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

SC and MR came up with the concept of Synchronized-PRNG and designed the hardware and simulation experiments. MR designed the 128-timer chipset and conducted the simulation and hardware experiments. SC provided supervision on all tasks. All authors contributed toward writing and proof-reading the manuscript. All authors contributed to the article and approved the submitted version.

Funding

This work was supported in part by the National Science Foundation grant EAGER-2237004.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Alimomeni, M., Safavi-Naini, R., and Sharifian, S. (2013). "A true random generator using human gameplay," in *Decision and Game Theory for Security*, eds S. K. Das, C. Nita-Rotaru, and M. Kantarcioglu (Cham: Springer International Publishing), 10–28.

Baker, S. B., Xiang, W., and Atkinson, I. (2017). Internet of things for smart healthcare: technologies, challenges, and opportunities. *IEEE Access* 5, 26521–26544. doi: 10.1109/ACCESS.2017.2775180

Bassham, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., Barker, E. B., et al. (2010). *Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications*. Technical report, Gaithersburg, MD, USA.

Bhattacharjee, K., and Das, S. (2022). A search for good pseudo-random number generators: Survey and empirical studies. *Comput. Sci. Rev.* 45, 100471. doi: 10.1016/j.cosrev.2022.100471

Eddy, M. (2022). *Is Your Car Key Fob Vulnerable to This Simple Replay Attack*? New York, NY: PCmag.

Golino, M. J. (2014). *System and Method of Secure Remote Authentication of Acquired Data*. Geneva: World Intellectual Property Organizations.

Günther, C. G. (1988). "Alternating step generators controlled by de bruijn sequences," in *Advances in Cryptology–EUROCRYPT' 87*, eds D. Chaum and W. L. Price (Berlin; Heidelberg: Springer Berlin Heidelberg), 5–14.

Henriques, M. S., and Vernekar, N. K. (2017). "Using symmetric and asymmetric cryptography to secure communication between devices in iot," in *2017 International Conference on IoT and Application (ICIOT)* (Nagapattinam), 1–4.

Hsueh, J.-C., and Chen, V. H.-C. (2019). An ultra-low voltage chaos-based true random number generator for iot applications. *Microelectronics J.* 87, 55–64. doi: 10.1016/j.mejo.2019.03.013

Johnston, D. (2021). *sp800_22_tests*. San Francisco, CA: Github.

Khazaei, S., Fischer, S., and Meier, W. (2007). "Reduced complexity attacks on the alternating step generator," in *Selected Areas in Cryptography*, eds C. Adams, A. Miri, and M. Wiener (Berlin; Heidelberg: Springer Berlin Heidelberg), 1–16.

Klein, A. (2013). *Linear Feedback Shift Registers*. London: Springer London.

Koopman, P. (2015). *Best crc Polynomials*. Pennsylvania: Carnegie Mellon university.

Kroese, D. P., and Rubinstein, R. Y. (2012). Monte carlo methods. *WIREs Comput. Stat*. 4, 48–58. doi: 10.1002/wics.194

Ma, Z., and Vandenbosch, G. A. E. (2012). "Impact of random number generators on the performance of particle swarm optimization in antenna design," in *2012 6th European Conference on Antennas and Propagation (EUCAP)* (Prague), 925–929.

Maghrebi, H., Portigliatti, T., and Prouff, E. (2016). "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering* (Cham: Springer), 3–26.

Massey, J. (1969). Shift-register synthesis and bch decoding. *IEEE Trans. Inf. Theory* 15, 122–127. doi: 10.1109/TIT.1969.1054260

Mehta, D., Rahman, M., Aono, K., and Chakrabartty, S. (2022). An adaptive synaptic array using fowler-nordheim dynamic analog memory. *Nat. Commun.* 13, 1670. doi: 10.1038/s41467-022-29320-6

Rahman, M., Bose, S., and Chakrabartty, S. (2023). On-device synaptic memory consolidation using fowler-nordheim quantum-tunneling. *Front. Neurosci*. 16, 1050585. doi: 10.3389/fnins.2022.1050585

Rahman, M., Zhou, L., and Chakrabartty, S. (2022). Spotkd: a protocol for symmetric key distribution over public channels using self-powered timekeeping devices. *IEEE Trans. Inf. Forensics Security* 17, 1159–1171. doi: 10.1109/TIFS.2022.3158089

Schindler, W., and Koç, Ç. K. (2009). *Random Number Generators for Cryptographic Applications*. Boston, MA: Springer U.S.

Staniec, K., and Kowal, M. (2020). On vulnerability of selected iot systems to radio jamming–a proposal of deployment practices. *Sensors* 20, 152. doi: 10.3390/s20216152

Sunar, B., and Koç, Ç. K. (2009). *True Random Number Generators for Cryptography*. Boston, MA: Springer U.S.

Wilber, S. A. (2017). *Synchronized True Random Number Generator*. US20180039485A1.

Yukonhiatou, C., Yoshihisa, T., Kawakami, T., Teranishi, Y., and Shimojo, S. (2020). A fast stream transaction system for real-time iot applications. *Internet Things* 11, 100182. doi: 10.1016/j.iot.2020.100182

Zhou, L., and Chakrabartty, S. (2017). "Self-powered timekeeping and synchronization using fowler-nordheim tunneling-based floating-gate integrators," *IEEE Transactions on Electron Devices* (IEEE), 1–7.

Zhou, L., Kondapalli, S. H., Aono, K., and Chakrabartty, S. (2019). Desynchronization of self-powered fn tunneling timers for trust verification of iot supply chain. *IEEE Internet Things J.* 6, 6537–6547. doi: 10.1109/JIOT.2019.2907930