SPoTKD: A Protocol for Symmetric Key Distribution over Public Channels Using Self-Powered Timekeeping Devices

Mustafizur Rahman, Liang Zhou, and Shantanu Chakrabartty, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a novel class of symmetric key distribution protocols that leverages basic security primitives offered by low-cost, hardware chipsets containing millions of synchronized self-powered timers. The keys are derived from the temporal dynamics of a physical, micro-scale time-keeping device which makes the keys immune to any potential side-channel attacks, malicious tampering, or snooping. Using the behavioral model of the self-powered timers, we first show that the derived key-strings can pass the randomness test as defined by the National Institute of Standards and Technology (NIST) suite. The key-strings are then used in two SPoTKD (Self-Powered Timer Key Distribution) protocols that exploit the timer's dynamics as one-way functions: (a) protocol 1 facilitates secure communications between a user and a remote Server; and (b) protocol 2 facilitates secure communications between two users. In this paper, we investigate the security of these protocols under standard model and against different adversarial attacks. Using Monte-Carlo simulations, we also investigate the robustness of these protocols in the presence of real-world operating conditions and propose error-correcting SPoTKD protocols to mitigate these noise-related artifacts.**

***Index Terms*—Key Exchange, Public-key Cryptography, Symmetric-key Cryptography, Self-Powered Timer, Quantum Key Distribution, Time-Synchronization.**

I. Introduction

SECURING information exchange with internet-of-things (IoTs), is becoming ever more important due to the proliferation of these platforms in domains ranging from infrastructure-IoTs [1] to medical-IoTs [2]. In one study [3] it is claimed that around 98% of the IoT data traffic is unencrypted and hence vulnerable to a data breach. Conventional data encryption techniques like RSA are too computationally prohibitive to be universally implemented on these low-resource platforms and reducing the computational complexity makes the approach vulnerable to quantum attacks. For instance, it is estimated in literature that a quantum computer with 8194 logical qubits using Shor's Algorithm would be able to break the Rivest-Shamir-Adleman(RSA) [4] system with a key size of 4096 bits in 229 hours while for Discrete log problem with a key size of 521 bits it would take 55 hours for a quantum computer with 4719 logical qubits, again using the Shor's Algorithm [5]. Symmetric key algorithms like Advanced Encryption Standard (AES-256) can be customized for IoT platforms and are considered to be secure against quantum attack [5], provided the security of the initial key-exchange can be guaranteed. Quantum key distribution(QKD) [6] which is based on the principles of quantum-mechanics, like quantum entanglement [7] or the no-cloning principle [8], [9] could be used to guarantee the security of the initial key-exchange. However, one of the major drawbacks of current state-of-the-art QKD systems is that they require dedicated and specialized peer-to-peer communication links [10], [11], [12], [13]. Not only do these links require careful maintenance and calibration to ensure quantum-coherence, but these systems are also expensive and not portable. Hence, current QKD systems cannot be scaled for internet-scale key distribution [14], [15] and communications involving lightweight IoT devices with resource constraints will still be vulnerable to quantum attacks.

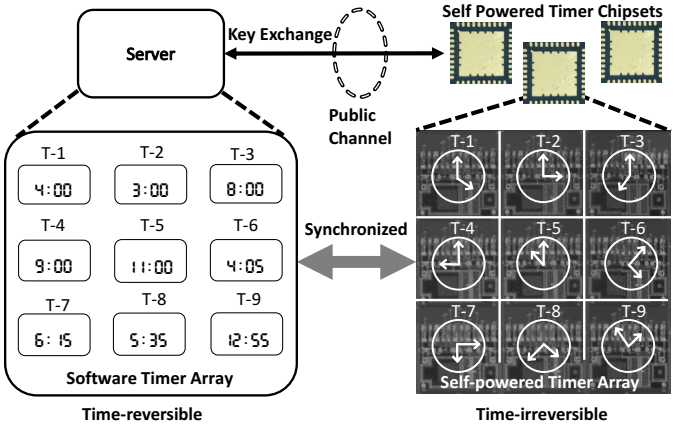

Figure 1. Framework underlying SPoTKD protocols: the synchronization and time-irreversibility of self-powered timers is exploited to implement one-way functions and facilitate secure key exchange over public channels.

In this paper, we propose a hardware-software Self-Powered Timer based Key distribution (SPoTKD) framework that does not require any modifications to the existing communication infrastructure, can be scaled to a large number of IoTs, and is potentially secure against quantum attacks. The approach relies on the trend that silicon-based chipsets with the capability of integrating billions of transistors and memory elements [16] can be manufactured in a large scale and at a low-cost [17]. If a physical feature on these chipsets could be exploited to implement a secure one-way function, then a hardware-software approach could be used to support key distribution

M. Rahman and S. Chakrabartty are with the Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, Missouri 63130, USA and L. Zhou is with Analog Devices Inc. All correspondences regarding this manuscript should be addressed to shantanu@wustl.edu.

This work is supported in part by a research grant from the National Science Foundation CNS-1646380

over public channels. In this paper, we propose one such method that exploits the synchronization capabilities and security features of our previously reported [18] self-powered timekeeping devices. The basic framework for SPoTKD is illustrated in Figure 1 where multiple identical copies of self-powered timer chipsets are openly distributed to all the users. Each of the timers on these chipsets is synchronized with its software clone running on a server. The key exchange between the server and the user is achieved based on this synchronization and time-evolution is used to implement a secure one-way function. It is to be noted that once the secret keys have been established and exchanged between the two parties, traditional symmetric cryptographic algorithms can be used for secure communications and user authentication [19].

The rest of this paper is organized as follows. Section II briefly describes other related protocols based on hardware-software based key distribution. Section III provides a brief background of the previously reported self-powered timers and their essential security features that have been exploited in the design of the SPoTKD protocols. In Section IV, we propose two SPoTKD protocols, one between a server and any user, and the other between two users. In Section V we analyze the security of the proposed protocols under various adversarial attacks. The robustness of protocol to operating and hardware artifacts have been analyzed in Section VI and in Section VII we introduce a variant of the protocol that uses error-correction codes to improve noise-robustness. We conclude the paper in Section VIII with discussions about the challenges and future directions.

II. Related Works

In literature, a few hardware-software key exchange methods have been proposed. In [20] a hardware-software public-key cryptography system for wireless networks was proposed based on Rabin's Scheme [21]. However, the security of Rabin's Scheme relies on the difficulty of factorizing large numbers, hence, it has similar vulnerabilities as the classical DH or RSA methods. Meanwhile, the one-way function (time irreversibility) implemented in SPoTKD is based on the principle of physics. Thereby SPoTKD does not suffer from such vulnerabilities. In [22] a hardware-software key exchange technique was proposed that exploited correlations across chaotic wavepackets in classic optical communications channels. However, the method still requires peer-to-peer connectivity between the users and hence has similar scaling disadvantages as QKD methods. On the other hand, SPoTKD uses silicon-based chipsets containing self-powered timers. Therefore, SPoTKD has the advantage against such key distribution methods for platforms with low computational resources. The hardware-software approach proposed in [23] used chaos synchronization to distribute random keys over public channels. However, due to the lack of reliable synchronization, this approach incurs significant errors during decryption. Recently, Physical Unclonable Function(PUF) based hardware-based encryption key distribution has been proposed. A specific variant of this technique, described in [24] as Public Physical Unclonable Function(PPUF) has been used for public-key cryptography and leverages the difficulty of accessing physical information stored on chipsets. However, in PPUF the stored information is static in nature and hence is potentially vulnerable to machine learning attacks [25], [26]. Whereas in SPoTKD the keys are derived from dynamic information that changes with time.

III. Self-powered Timer Security Primitives

The SPoTKD protocol exploits the physical features of self-powered timers to ensure the security of the key exchange. The design and the operating principle of self-powered timers have been previously reported in [18], [27]. In this section, we discuss the basic security primitives offered by the timer's physical response that will form the axiomatic core of the security analysis for SPoTKD that is presented later in this paper.

A. Self-powered timers are immune to power side-channel attacks

A simplified equivalent circuit model of the self-powered timer is shown in Fig 2(a) where a leakage-current J_{tunnel} is used to discharge a floating-gate capacitor C_T. Thus, once the floating-gate capacitor C_T is charged or programmed initially, no external power is required to drive the dynamics of the discharge process. The change in the floating-gate charge/voltage is monotonic with respect to the time elapsed and this feature has been previously used for time-keeping, synchronization, and authentication [18], [28]. For this work, the self-powered operation decouples the timer from the external power supply. This provides security against any power side-channel attack that might be aimed at gaining knowledge about the current state of the timer by observing fluctuations in the supply-current.

B. Self-powered timers are immune to electromagnetic side-channel attacks

The leakage current J_{tunnel} in the self-powered timer is implemented using Fowler-Nordheim(FN) tunneling of electrons through a thin gate-oxide barrier. In [27], we have shown that the operation of the timers is robust even when the FN tunneling current is as low as one electron per second (or less than an attoampere). From a security point of view, the low tunneling current practically eliminates any electromagnetic (EM) emission and hence any EM side-channels. Also, any unauthorized attempt to access the timer-state using an EM probe desynchronizes or destroys the state of the timer.

C. Dynamics of the self-powered timers can be synchronized

One of the essential attributes of the timer that is important for the realization of the SPoTKD protocol is that the timer's temporal responses can be synchronized not only with respect to each other but also to a well-defined behavioral (or software) model. For this work, we use a specific form of the timer behavioral model that is given by

$$I_{timer}(t) = p_3 \exp\Big[-\frac{p_2}{\log(p_1 t + p_0)}\Big]. \tag{1}$$

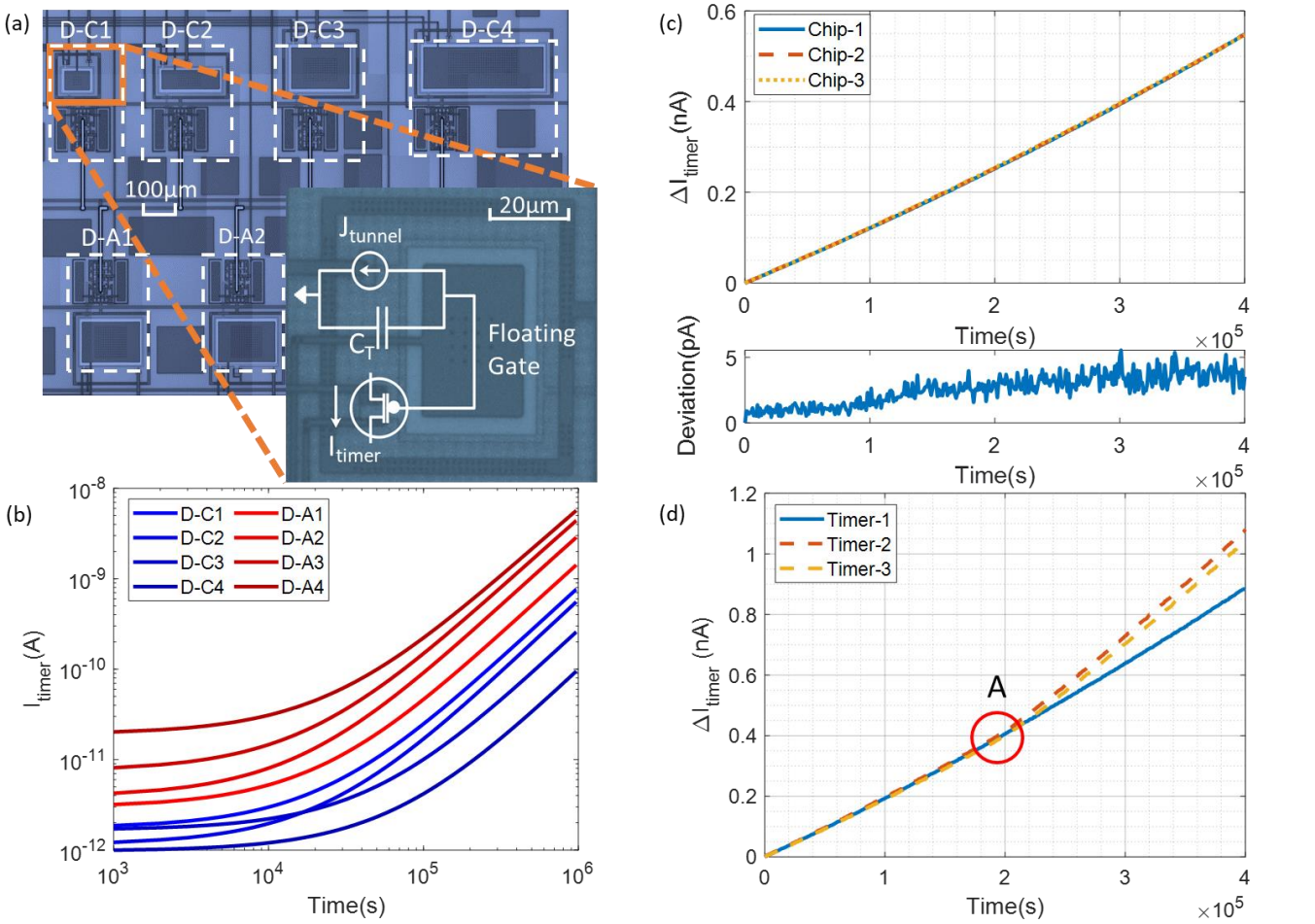

Figure 2. (a) Micrographs of self-powered timers (labeled as D-C1, D-C2, D-C3, D-C4) with different form factors and features that determine the parameters of the timer behavioral model in equation 1. The equivalent circuit model for a single timer along with the readout circuit is shown in the inset. (b) The temporal responses measured using these timers for (a) different initialization conditions. (c) Synchronization of a timer's temporal response with same form factors across multiple chipsets after the initial transient response. (d) Desynchronizing the temporal response of different timers by coupling an external source of energy into two of the timers at the time-instant denoted by A.

where $I_{timer}(t)$ is the current measured at time instant t quantifying the state of the timer. The current is measured using a read-out metal-oxide-semiconductor field-effect transistor (MOSFET) whose gate is coupled to the floating-gate, as shown in Fig. 2(a). The behavioral model in equation 1 assumes that the read-out transistor is biased in a specific regime, details of which can be found in the derivation of the behavioral model in the Supplemental Material. The tuple $\overline{P} = [p_0, p_1, p_2, p_3]$ in equation (1) are the timer parameters that are determined by the device form factors and the device initialization conditions. Figure 2(a) shows an example of a system-on-chip implementation that integrates different timer structures with varying form-factors. The responses of these timers with different initialization conditions are presented in Figure 2(b) which shows that the temporal dynamics of each timer is unique and is determined by the tuple $\overline{P}$. We have previously shown that for a fixed set of timer parameters $\overline{P}$ the mathematical model in equation 1 can capture the temporal behavior of the timer for more than a year with an accuracy of greater than 0.5% [27]. This is shown in Figure 2(c), where the timers with the same form-factor but integrated on different chipsets remain synchronized with each other. The deviation between the timer's responses is in the range of pico-amperes and this synchronization error can be attributed to the measurement noise and not to the synchronization error. For the SPoTKD protocol, the synchronization between the behavioral model (or software timer) and the hardware timers will be used for key exchange. The key exchange will exploit the asymmetry between the software timers and hardware timers where that the hardware timer cannot be rewound (or time-irreversible) whereas its software clone can be rewound to any previous time instant. This asymmetry is exploited as a one-way function for securing the SPoTKD protocol. Note that the parameters $\overline{P}$ which determine the dynamics of each timer, are never revealed publicly and therefore functions as a private key in our protocol. Later in Section V we show that it is practically impossible to extract these parameters from measurements on the hardware timer itself.

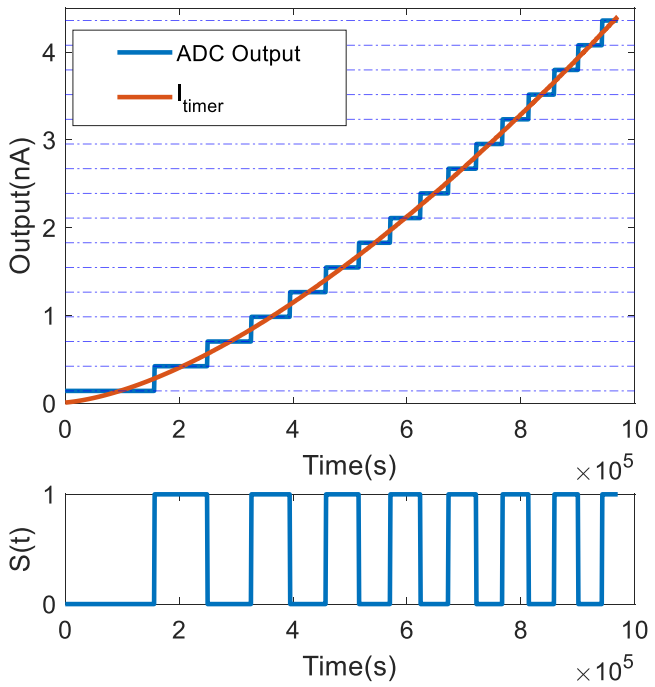

Figure 3. Dynamic binary state $s(t)$ of a timer generated after the analog current is read-out with an ADC. Illustration here shows the state $s(t)$ corresponding to a 4-bit ADC.

D. Self-powered timers are designed for one-time read and tamper-resistant

In [27] we showed that the synchronization between the timers could be broken by injecting an external signal into the floating-gate. This is demonstrated in Figure 2(d) where three timers (with similar form factors) are synchronized with respect to each other till time-instant 'A'. Then at time-instant 'A' an external energy-source is coupled to timers 2 and 3 (in this case using capacitive coupling). As a result, these timers become de-synchronized from each other. We will use this controlled de-synchronization feature to intentionally destroy the dynamical state information stored on each timer once its state has been accessed. Thus, each of the timers can only be used once to generate the key-string after which the state of the timer is destroyed (or desynchronized). Note, the desynchronization of the timer can also result when the timer is unintentionally probed (using hardware delamination or using electromagnetic probing). This feature makes the basic timer tamper-resistant.

E. Bit generation using self-powered timer

We will assume that the state of the self-powered timer can be measured using an on-chip analog-to-digital converter(ADC) where the least-significant-bit (LSB) represents a modulo-2 measurement of the timer value. Denoting the binary state $s(t) \in \{0,1\}$ of the timer as the LSB obtained after the $I_{timer}(t)$ is measured at a time-instant t, then $s(t)$ can be expressed as

$$s(t) = \lfloor \frac{I_{timer}(t)}{\delta} \rfloor \mod 2 \quad (2)$$

where δ is the resolution of the ADC. This is illustrated in Figure 3 where a 4-bit ADC is used to measure $I_{timer}(t)$ to generate the LSB or $s(t)$. For the protocols proposed in this paper, we will also assume that once the binary state of a timer is measured, its state is destroyed through a process of desynchronization, as described in the section III-D. This implies that each timer can only be used once to generate a single bit '$s(t)$' at a given time t for key-generation.

F. Summary of hardware security primitives offered by self-powered timers

Here we summarize the security primitives that is offered by self-powered timers and will serve as axioms for the proposed SPoTKD protocol:

- SP1: It is practically impossible to access any information about the secret parameters or the state of the timer using side-channels (power or electromagnetic) attacks.
- SP2: The temporal behavior of each timer is unique and are determined by the timer's secret parameter tuple $\overline{P}$.
- SP3: The binary state of a timer $s(t)$ as defined in equation 2 is dynamic in nature and changes with time. As a result, the state of a timer is unpredictable without knowledge about the secret parameters of the timer.
- SP4: The hardware chipsets are designed in such a manner so as users are limited to only the output of the chipsets after following specific protocol (discussed in Section IV). Any attempt to snooping on the hardware chipsets otherwise would result in a destruction of the information embedded on the timers.
- SP5: The state of each timer in a chipset can only be accessed once, after which the state is erased or destroyed.
- SP6: The number of hardware chipsets that are available at any given instance of time is finite.

IV. SPoTKD Protocol

The basic SPoTKD protocol is shown in Figure 4. A server creates multiple replicas of chipsets each of which integrates a set $\mathcal{T}$ of $C \in \mathbb{Z}^+$ timers. Each timer in the set is assumed to be initialized according to a parameter tuple $\overline{P}_i$, where $1 \leq i \leq C$, as defined in equation (1). Note that some of the parameters (initial charge on the floating-gate) in the tuple are programmed by the server and some of the parameters (device form-factor) are fixed post-fabrication. Also, note that only the server has access to this information and is kept secret from the users. These identically programmed chipsets are then distributed to all the users over a public distribution channel, as shown in Fig. 1. When an intended user wishes to communicate with the server, they arbitrarily choose to measure the binary states of two sets of timers which will be referred to as 'hash' timers and 'key' timers. The objective is to use the G 'hash' timers and N 'key' timers to generate an N bit long binary key $\mathbf{K_B} \in \{0,1\}^N$. To achieve this the outputs of G randomly chosen hash timers $s_{H_1}(t), .., s_{H_G}(t)$, $1 \leq H_1, .., H_G \leq C$ measured at time instant t are XOR-ed with each other to generate a single bit $X(t)$ according to

$$X(t) = s_{H_1}(t) \oplus s_{H_2}(t) \oplus s_{H_3}(t) ..... \oplus s_{H_G}(t) \quad (3)$$

Note that the time instant $t \in \mathbb{R}^+$ is referenced according to a universal standard time. The key bits $Q_L(t), L = 1, .., N$ are

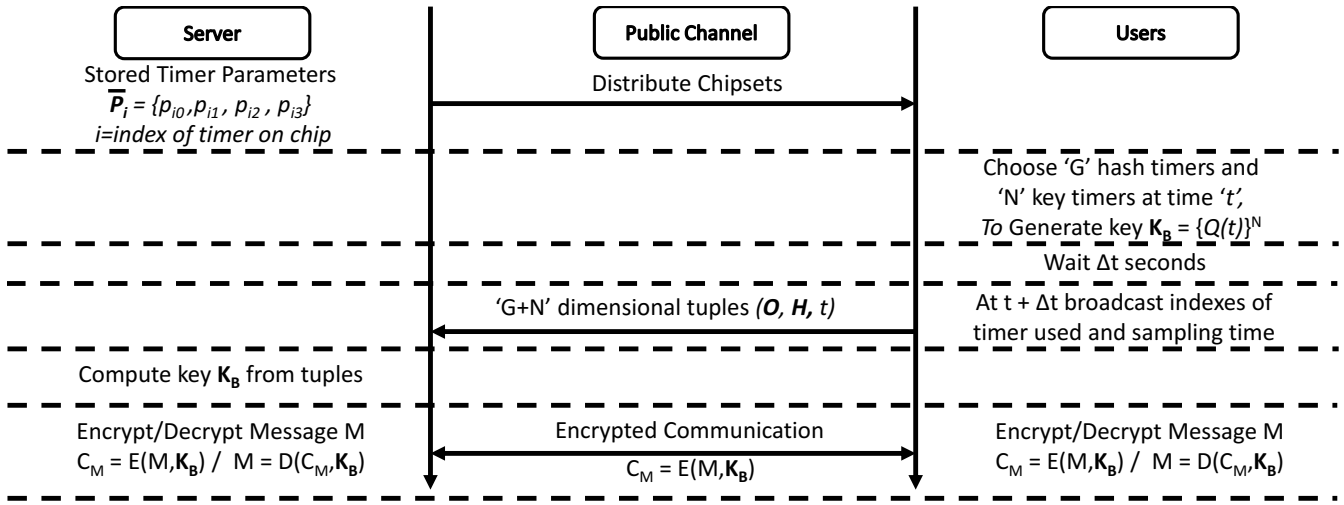

Figure 4. Basic SPoTKD protocol between the server and a user. Here $E(M, \mathbf{K_B})$ represents an encryption function where message M is encrypted with key $\mathbf{K_B}$, $D(C_M, \mathbf{K_B})$ is the decrypting function where C_M is the cipher text being deciphered with key $\mathbf{K_B}$.

then generated at time t by XOR-ing the binary states of each of the 'key' timers $s_{O_1}(t), .., s_{O_N}(t); 1 \leq O_1, .., O_N \leq C$ with $X(t)$ according to

$$Q_L(t) = s_{O_L}(t) \oplus X(t) \tag{4}$$

to generate $\mathbf{K_B} = \{Q_L\}^N$.Note that since the state of each of the timers can only be accessed once, the 'hash' and the 'key' timers need to be different, namely $\{O_1, .., O_N\} \cap \{H_1, .., H_G\} = \emptyset$. Also, note that the user can only access the N bit key string $\{Q_L\}^N$ and not the binary states of the 'key' timers or $X(t)$ from the hardware chipsets.

In the next step of the SPoTKD protocol, as shown in Figure 4, the user waits for a random time-duration Δt seconds after which they broadcast a $G+N$ dimensional tuple $(\mathbf{O}, \mathbf{H}, t)$ over the public channel. Note that here t indicates the time at which the G 'hash' and N 'key' timers were accessed and only the indices of the timers are broadcasted (and not measured output). The server then uses the tuples $(\mathbf{O}, \mathbf{H}, t)$ and its knowledge of the 'secret' parameters $\overline{P_i}, 1 \leq i \leq C$ to decipher the binary states of all these timers and compute the key $\mathbf{K_B}$ completing the key exchange.

The SPoTKD protocol shown in Figure 4 is suitable for communicating between a user and a server that owns and initializes all the timer chipsets. However, key exchange be-

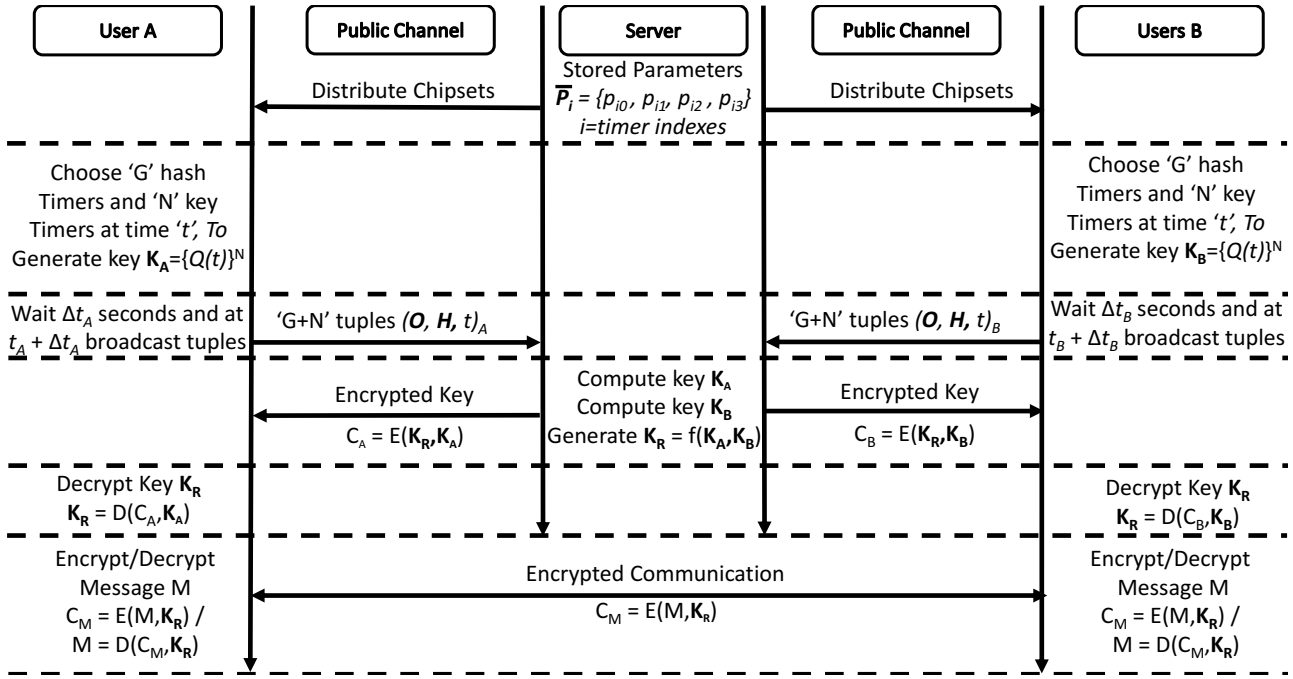

Figure 5. SPoTKD protocol for exchanging keys between two users with server acting as a trusted third party.

tween two users can also be facilitated with the help of the server acting as a trusted third party, as shown in Figure 5. In this protocol, both the users broadcast their tuples $(\mathbf{O}, \mathbf{H}, t)_A$ and $(\mathbf{O}, \mathbf{H}, t)_B$ over a public channel. The server deciphers both keys, $\mathbf{K_A}$ and $\mathbf{K_B}$ according to previous protocol. The server then generates a new key $\mathbf{K_R}$ which is a function of the keys $\mathbf{K_A}$ and $\mathbf{K_B}$. This function $f : \{0,1\}^{2N} \rightarrow \{0,1\}^N$ is decided by the server and can be any mathematical operation ranging anything from multiplication to complex hashing. This operation is never revealed and changed for every session. The server then sends cipher texts $C_A = E(\mathbf{K_R}, \mathbf{K_A})$ to user A and $C_B = E(\mathbf{K_R}, \mathbf{K_B})$ to user B containing the key $\mathbf{K_R}$ encrypted using $\mathbf{K_A}$ and $\mathbf{K_B}$ respectively. The users can decrypt the cipher text to know the secret key $\mathbf{K_R}$. For further communication, each user uses this key $\mathbf{K_R}$ to encrypt and decrypt their messages with each other. Since all keys are randomly generated and have never been used before then anyone intercepting the cipher text will not gain any information regarding the secret key being used. Note that in this protocol the users do not need to match either the timers they used in the chip or the time at which they will generate their respective keys. They only need to agree upon their time of communication and can generate their keys beforehand individually. In order to update any new session key between two users, the users would need to use a new set of timers and follow the same protocol for exchanging keys with the server acting as the trusted third party.

V. Security and Performance Analysis

According to the recommendation by National Institute of standards and technology (NIST), a 256-bit key is sufficient for symmetric key algorithms to be secure [29] even in the presence of a quantum computer. While using Grover's algorithm, a quantum computer with 6681 logical qubits and approximately 3.36×10^7 physical qubits would require around 2.29×10^{32} years for a brute force search attack on AES-GCM cryptosystem with a 256-bit key size [5]. Hence, for the rest of the paper, we will show test results corresponding to 256-bit keys, generated from $G = 128$ hash timers and $N = 256$ key timers for all analysis purposes. Note that, the number of hash timers used in key generation determines the complexity of the key generation. We will show that $G = 128$ hash timers are sufficient for the protocol to be secure. Increasing the number of hash timers would further increase the complexity but would come at the cost of noise robustness. Since the scope of this work is only to propose a secure key exchange protocol that can be used for symmetric-key encryption schemes, our security analysis will only focus on showing that the key exchange protocol is quantum secure.

For our first analysis, we consider the scenario where an attacker simply attempts to guess the key without any information about the key generation system. As long as the keys that are generated from the timers are completely random in nature, the attacker will not gain any unfair advantage. So we tested the secret keys generated according to the SPoTKD protocol described in Section IV with the NIST test suite for checking randomness of bit stream [30]. The suite usually

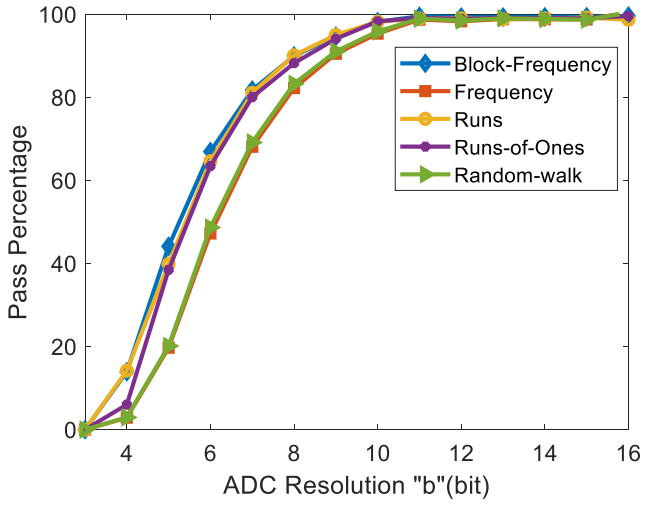

Figure 6. Pass percentage obtained using the NIST randomness test suite applied to the keys generated using the SPoTKD protocol, as a function of the resolution b of the ADC used to measure the state of the 'key' timers.

consists of fifteen different tests to measure the randomness in a certain bitstream. However, a few of these tests require a large sequence of bitstream which does not apply for a length of 256-bit keys. Therefore, in our analysis we show the test result for 5 of the suitable tests. The binary states for the hash timers were always measured with an 11-bit ADC irrespective of the key timers. This was performed to ensure better noise robustness. If a higher resolution ADC was used to sample the hash timers, then the noise robustness of the protocol would decrease (discussed in Section VI). Using Monte Carlo simulations, we sampled 10^6 keys at random time instances using a b-bit ADC (or 2^b-1 level quantizer) for the key timers. We extracted the parameter tuples $\overline{P} = [p_0, p_1, p_2, p_3]$ from the timer responses shown in Figure 2(b) and then randomized within the range of these actual hardware parameters to represent unique timers in our simulations. This ensures that each timer used in the simulation can actually be realized in hardware chipsets. Figure 6 shows the pass percentage, i.e. the percentage of keys from the 10^6 samples that passed the test, as the resolution 'b' of the ADC is varied for the key timers. We can observe from the plots that for large values of 'b', almost all the generated keys pass the test. The randomness degrades for ADC resolution less than 8 bits showing that a 9-bit ADC for the key timers should be sufficient to generate high-quality keys. This shows that keys derived from the timer responses are completely random in nature and any attempt to guess the key would result in a brute force search which is the same as breaking the AES-256 encryption scheme discussed above. Moreover, it also means that the binary state of each timer is uncorrelated with other timers and an attacker cannot simply sample the binary states of any one timer and can predict what other timers' response would be at any given point in time. This is in accordance with the axioms SP2 and SP3 discussed in Section III.

Next, we consider the information that is available to an attacker and investigate whether he can gain any advantage while predicting the key-string with the information available

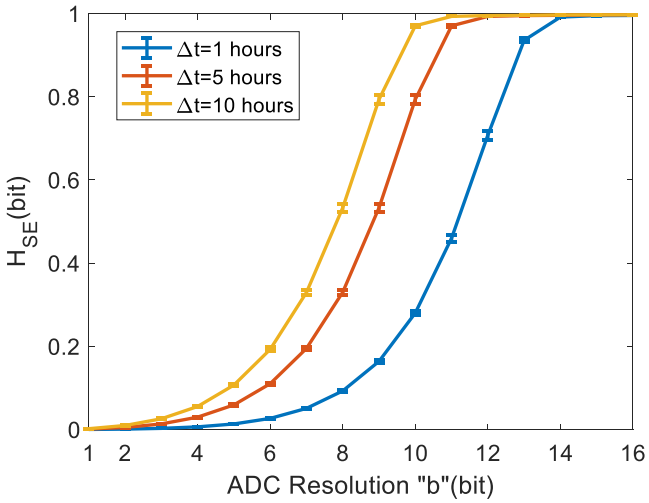

Figure 7. Uncertainty per bit measured for three different waiting periods Δt as a function of the resolution b of the ADC used for measuring the state of the 'key' timers.

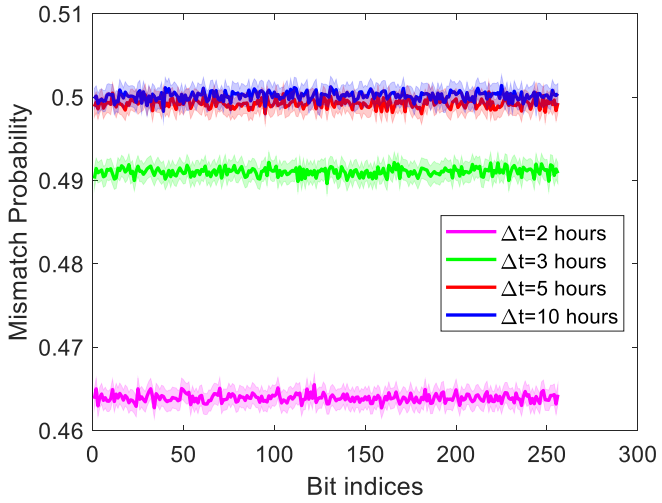

Figure 8. Probability that the binary states of a timers used in key generation has changed after the waiting period Δt hours. Here the resolution of the ADC used for key generation is b=12-bits. The variance across different Monte-carlo trials are highlighted by the shaded region.

to him. So, here we note down all the information and resources about the key exchange framework that is potentially available to an attacker:

- I1: We assume that the attacker can passively eavesdrop on the communications over the public channel. This means that the attacker would know which timers were used for a particular key-string.
- I2: We also assume that the attacker has access to the hardware chipsets.
- I3: The attacker knows the underlying principle of the timers' behavior and other logistics of the protocol as described in this work.
- I4: The attacker has access to a fully functioning quantum computer.

Now considering I1 and I2, a potential attack could be launched by the adversary where they sample the timers on their copy of the chipset as soon as the user broadcasts a tuple $(\mathbf{O}, \mathbf{H}, t)$ over the public channel. However, the key that the attacker generates will be at a time instant $t+\Delta t$, where Δt is the time that the user waits after they have generated the key. Since the timer values are dynamic in nature, the key generated by the attacker $\mathbf{K_E}$ will be different from the key generated by the user $\mathbf{K_B}$. To quantify the disparity between the keys, we use Shannon information entropy to measure how much information can the attacker gain about $\mathbf{K_B}$ using their own key $\mathbf{K_E}$. The average Shannon information entropy contained in each bit generated by the attacker can be expressed as

$$H_{SE} = -d\log_2 d - (1-d)\log_2(1-d) \tag{5}$$

where d is the average difference in bits between $\mathbf{K_B}$ and $\mathbf{K_E}$. The parameter H_{SE} quantifies the uncertainty of the attacker for every bit of the key $\mathbf{K_B}$ that he or she tries to predict using $\mathbf{K_E}$. When $d = 0$ i.e. the attacker generates the same key as the user, the information entropy of the attacker is zero, this is because the attacker can predict the key with perfect certainty. A similar argument can be made for the other extreme scenario, when $d = 1$, as the attacker can simply invert each bit that he or she generates and produce $\mathbf{K_B}$. The entropy H_{SE} is also equal to 0 in this case. On the other hand, when $d = 0.5$ exactly half of the bits of $\mathbf{K_E}$ do not match with $\mathbf{K_B}$. This means that if the attacker were to randomly guess all the key-bits they would, on average, end up with the same number of matched bits. Therefore, the attacker has 1 bit of uncertainty for every bit generated and zero information gain on the key. The entropy H_{SE} thus takes the maximum value of 1 in this case.

In order to mimic such a kind of attack we sampled a set of timers and generated keys at random time instances, representing the user's key, and also sampled the same set of timers at a later instant, which represents the attacker's key. After that, we calculated the entropy for each sample. Figure 7 shows the average uncertainty per bit generated by the attacker when he or she samples the same timer array used by the user. We can observe from the figure that for keys generated with high-resolution ADC, the attacker has almost 1 bit of uncertainty per bit. This means that the attacker is unable to gain any information about the user's key from sampling their own timer chipset. The overall trend for the curves with the same wait period (which corresponds to Δt) can be explained by the fact that at higher resolution, the LSB contains minimum information about the whole dynamic response of the timer. Moreover, the LSB changes much more frequently, and therefore key generated using LSB is more difficult to predict for the same waiting period. It gets increasingly easier to predict as the resolution of the ADC is decreased since the LSB changes slowly and sampling yields more information.

The uncertainty can be increased for a lower-resolution ADC by increasing the waiting period Δt which is shown in Figure 7 where the curve shifts towards the left as we increase Δt. This is because as Δt is increased, the probability that an ADC bit has changed will also increase, thereby sampling the bits will not provide any useful information.

However, average Shannon information entropy H_{SE} is agnostic of the position of the mismatched bits. For instance, one pathological case could be that always the first half of the key-string obtained by the attacker is mismatched while the second half always matches with the true key-string. In this scenario, HSE would still be 1, but the attacker can easily guess the correct key-string. In our next analysis, we show that the probability of such a case is negligible (practically does not exist). Using Monte Carlo simulations with ADC resolution b=12-bits, we counted how many times each of the key-bits in the 256-bit key string gets mismatched among all the iteration and calculated the probability of mismatch for each bit index. Figure 8 shows the probability of mismatch for each bit index after different waiting periods. We can observe that as the waiting period increases each bit index has an approximately equal probability of 0.5 for being mismatched. This shows that there is no bias with respect to the positioning of the mismatched bits and each key bit generated by the attacker has an equal probability of being correct or incorrect which is the same as purely guessing. For a lower waiting period the probability of mismatch decreases for all the bit indices which is in accordance with our previous analysis, but the mismatch probability is approximately the same irrespective of the bit position. Thus, as long as a reasonable resolution ADC is used for measuring the state of the timer and the waiting period is large enough, the attacker will not be able to predict as to what key string was generated by a user. Therefore, I1 and I2 do not reveal any information about the secret key and the attacker would still need to resort to brute force search for a successful attack.

So far we have shown that the key exchange protocol is secure based on the facts that the keys used are completely random in nature and from the public information available during the key exchange the attacker can not gain any information about the random keys. Next, we consider I3 available to an attacker and investigate whether they could predict the keys by using their knowledge about the timer behavioral (or software) model. However, since they do not have access to the timer initialization parameters $\overline{P_i}$, they cannot use the public information $(\mathbf{O}, \mathbf{H}, t)$ to decipher the states $s_{O_l}(t)$. Also, the attacker is unable to rewind the hardware timer on their copy of the chipset to measure the states $s_{O_l}(t)$ going back in time. Therefore, the only way to predict $\mathbf{K_B}$ would be to solve equation 4 for each bit of the key for finding the secret parameters $\overline{P_1}, \overline{P_2}...\overline{P_N}$. In the next set of analysis, we will show that it is practically impossible to find the secret parameters from the hardware chipsets themselves.

First, we consider equation 1 where the parameters could be regressed if a timer is sampled multiple times to measure $I_{timer}(t)$ at different time instances. However, this is only true if the attacker can get access to the precise value of $I_{timer}(t)$. From equation 2 we observe that the binary state of the timer only provides a single bit of information about $I_{timer}(t)$. Moreover, axiom SP4 dictates that even the single bit of information about $I_{timer}(t)$ is XOR-ed with other G hash timers' binary states. The attacker has only access to the XOR-ed output due to the manner in which hardware chipsets are designed. Therefore each bit of key-string the attacker samples from the hardware chipsets will be derived from $G+1$ timers. Note that there is no analytical solution for equation 4 so the attacker will have to resort to a brute-force numerical search. We now show how the SPoTKD protocol is secure against such attacks under the standard model.

Claim 1. The SPoTKD protocol is secure under the standard model.

Proof. Each key bit $Q_L(t)$ is derived from the temporal responses of $G+1$ timers where G is the number of hash timers used in key generation. Now, the temporal response of each timer is determined by the secret parameter tuples $\overline{P} = [p_0, p_1, p_2, p_3]$. Therefore, each key bit, in turn, is determined by $G+1$ tuples of $\overline{P}$. We define p_{Total} as the total number of parameters from which each bit is derived which is given by

$$p_{Total} = 4(G+1) \tag{6}$$

This means that the search space would be a matrix with p_{Total} dimensions. Now, let R be the range of possible values for each of the p_{Total} parameters. Then the total number of elements in the matrix i.e. the total search space SP_{Total} would be given by

$$SP_{Total} = R^{4(G+1)} \tag{7}$$

Even though the parameters $\overline{P}$ are determined by the timer initialization conditions and timer form factors, they are calibration parameters. Assuming a double-precision floating-point for the parameters implies that $R = 2^{63}$. This yields

$$SP_{Total} = 2^{252(G+1)} \tag{8}$$

For $G = 128$ hash timers (which was used in our simulations for generating the key string) this would result to a search space of 2^{32508} possible combinations. Therefore, an attacker employing a brute-force search strategy would require 2^{32508} bits of storage, which is prohibitively large. Moreover, even if the attacker uses the fastest computer in the world [31], which can perform 10^{19} computations per second, it will take them approximately 2^{32444} seconds, or 2^{32419} years to search the entire space. Since we assumed that the attacker is only constrained by the computational/storage resources and time available to them, hence, under the standard model, the SPoTKD protocol is secure. ■

Next, we consider I4 where we assume that the attacker has access to a quantum computer with large enough storage space and computational resources to search the aforementioned solution space in a reasonable amount of time. In this analysis, we show that our protocol remains secure if we impose a physical constraint that limits the number of hardware chips that the attacker can use for measurement.

Claim 2. The SPoTKD protocol is resistant to quantum attacks.

Proof. Equation (4) has no unique solution and since the parameters are randomly chosen by the server, every solution within the search space is equally likely to be the correct one. The only way to eliminate possible combinations from the solution set would be to sample each hardware timer at

Table I: Performance Comparison between SPoTKD and other state-of-the-art key exchange protocol

Protocol	**Key Length** (bits)	**Security Strength** (bits)	**Computational Cost** (no. of cycles)	**Scalability**
PPUF [24]	1024	112	$\mathcal{O}(10^{16})$	High
QKD [6]	256	256	$\mathcal{O}(10^{4})$	Low
RSA [4]	3072	128	$\mathcal{O}(10^{7})$	High
SpoTKD	**256**	**256**	$\mathcal{O}(\mathbf{10^{2}})$	**High**

multiple time instances and solve equation 4 repeatedly. Since equation 2 is symmetric the expected size of the solution set, denoted as $\mathbb{E}(SP_J)$, after each sampling reduces by

$$\begin{aligned}\mathbb{E}(SP_J) &= \frac{SP_{Total}}{2^J} \\ &= \frac{2^{252(G+1)}}{2^J}\end{aligned} \tag{9}$$

where $J \in \mathbb{Z}^+$ indicates the number of samples. This means that if the attacker can sample each timer enough number of times, they can find out the initialization parameter $\overline{P}$. However since the timers are designed for one-time read (Axiom SP5 in section III), the attacker is unable to make multiple measurements on a timer using the same chipset. For each measurement, the attacker would therefore require a new chipset. Thus, there is an upper bound to the number of measurements that an attacker can perform, which is the total number of chipsets C_{Total} available. Therefore we have

$$J \leq C_{Total} \tag{10}$$

Now if we constrain the total number of chipsets C_{Total} according to

$$C_{Total} < 252(G+1) \tag{11}$$

then the attacker would still be unable to find the unique solution to equation 4 since

$$\mathbb{E}(SP_J) > 2 \qquad \forall J \tag{12}$$

Note that, the constraint here for an attacker is not the computational power available to them but rather the physical resources they can acquire. Thus, the key exchange protocol is resistant to quantum attacks. ■

In the next set of analysis we want to show how the proposed key exchange protocol is secure against most popular kind of attacks.

Claim 3. The proposed protocol is secure against man-in-the-middle attacks.

Proof. During the SPoTKD protocol, a user publicly broadcasts the tuples $(\mathbf{O}, \mathbf{H}, t)$ indicating the timer indexes the user sampled along with the time at which they were sampled. For an attacker to successfully impersonate the server, they will need to know the secret timer parameters $\overline{P}$, which is never revealed during any phase of the protocol. Also, our previous analysis shows that it is practically impossible to find out these parameters using brute-force search. Note that all the publicly distributed chipsets store the same information on the timers and authentication is carried out only after the server and user have established a secure channel subsequent to a successful key exchange. Thus, the attacker cannot impersonate any user. ■

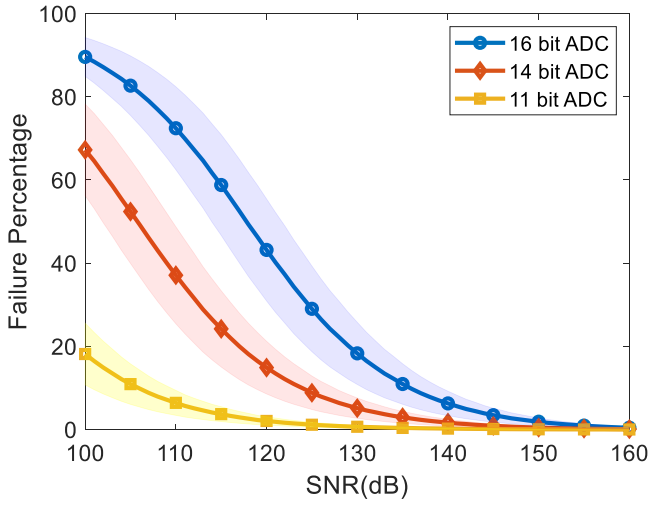

Figure 9. Improvement in noise-robustness of the SPoTKD protocol when the resolution b of the ADC used for measuring the state of the 'key' timers is decreased. The variance across different Monte-carlo trials are highlighted by the shaded region.

Claim 4. The proposed protocol is secure against replay attacks.

Proof. Once a set of timers is used for key exchange, they are desynchronized with respect to the server's model (Axiom SP5 in section III). Thus, during every session, a new set of timers is used to exchange keys. This means that a new key is generated for every new session. Also, during the key exchange protocol, the measured states of the timers are never made public. Therefore, the attacker cannot use any information from previous sessions to their advantage. This implies that the SPoTKD protocol is secure against replay attacks. ■

Claim 5. SPoTKD protocol is secure against backward and forward traceability attacks.

Proof. In our protocol, the keys generated are random in nature as shown in figure 6 that are not predictable. Also, each key is used only once. Therefore the key exchange at session instance SS_a can not be inferred from other keys at any other session SS_b, where $a \neq b$. Moreover, we have shown in the previous claims that inferring any knowledge about the secret parameters is also practically impossible. Therefore, the SPoTKD protocol is immune to forward or backward traceability attacks. ■

Claim 6. SPoTKD protocol is resistant to de-synchronization attacks.

Proof. The robustness of the timer response ensures that the dynamics of the hardware timer remain synchronized with its software model on the server. According to Axioms SP1-SP4

in section III-F, the timer's dynamic response on any user's chip cannot be programmed or altered by the attacker unless and until the attacker gets access to the chip physically. In such a case where the user suspects that his or her chip may have been compromised physically by an attacker, the user can simply discard the chip and procure a new one, since all the chipsets have the same information that is stored. Thus, the protocol is resistant to de-synchronization attacks. ■

In addition, the construction, operating principle and inherent security of the quantum-tunneling device i.e. the self-powered timer [18] also prevent the attacker to probe the state of the timer by using any side-channel (power or electromagnetic) without affecting the state of the timer (Axioms SP1-SP6 in section III-F). Therefore, in this regard, the timer chipset emulates a quantum communication channel [19], but using an analog dynamical system that is secure against any side-channel attacks.

We have evaluated the performance of our proposed protocol with similar hardware-software based key exchange protocols such as PPUF [24] and some state-of-the-art key exchange protocols such as RSA [4] that are currently being used. The comparison is summarized in Table 1 with respect to criteria such as key length, security strength, computational cost, and scalability. Here security strength measures the number of trials required to brute-force a key irrespective of the key length. A 128-bit security means 2^{128} trials to break the protocol. We also compared the computational resources required to perform a single key exchange in terms of the number of computation cycles. And finally, scalability indicates the ease at which the key exchange protocol can accommodate large of number of users. Since our goal is to provide secure key exchange among a large number of users using low resources, these features are extremely important to evaluate and compare different designs.

We start by evaluating the security strength of each protocol. For PPUF using 1024 bit key, an attacker needs to perform $1.7x10^{29}$ cycles of simulation on average to find the secret key [24]. Accounting for overhead computation this roughly translates to a 112-bit security. According to NIST 2020 recommendations, RSA requires a key length of 3072-bits to achieve a security strength of 128-bit. Now, since both QKD and SPoTKD use symmetric key encryption (AES), a 256-bit key length corresponds to a security strength of 256-bit. Due to the use of a large key size, both PPUF and RSA are computationally expensive. The PPUF based key exchange protocol requires approximately 10^{16} cycles of computation [24] and RSA requires $\mathcal{O}(10^7)$ computational cycles [32]. Even though QKD uses a much smaller key-string, additional computation needs to be performed for the error reconciliation protocol. The computational complexity is of order $\mathcal{O}(10^4)$ for a 256-bit key using common error-correcting code [33]. Meanwhile, for the basic SPoTKD the user needs to simply measure the state of the timers once and perform $G = 128$ bit-wise XOR from the hash timers to generate the bit $X(t)$. This can be done in $log_2(G)$ computational cycles. After that, the outputs of the N key timers are XOR-ed with the bit $X(t)$ in N cycles. In this regard, our protocol is by far the most efficient. If error correcting SPoTKD is used(discussed in Section VII), the computational cost will be similar to that of QKD. In addition, we have shown in our analysis that our protocol is resistant to quantum attacks, similar to QKD. In comparison, however, QKD is expensive and in its current state is not portable or scalable to support a large number of users. On the other hand, our protocol is based on silicon fabrication technology which is relatively inexpensive at a production scale and the fabricated chipsets can be easily distributed to millions of users.

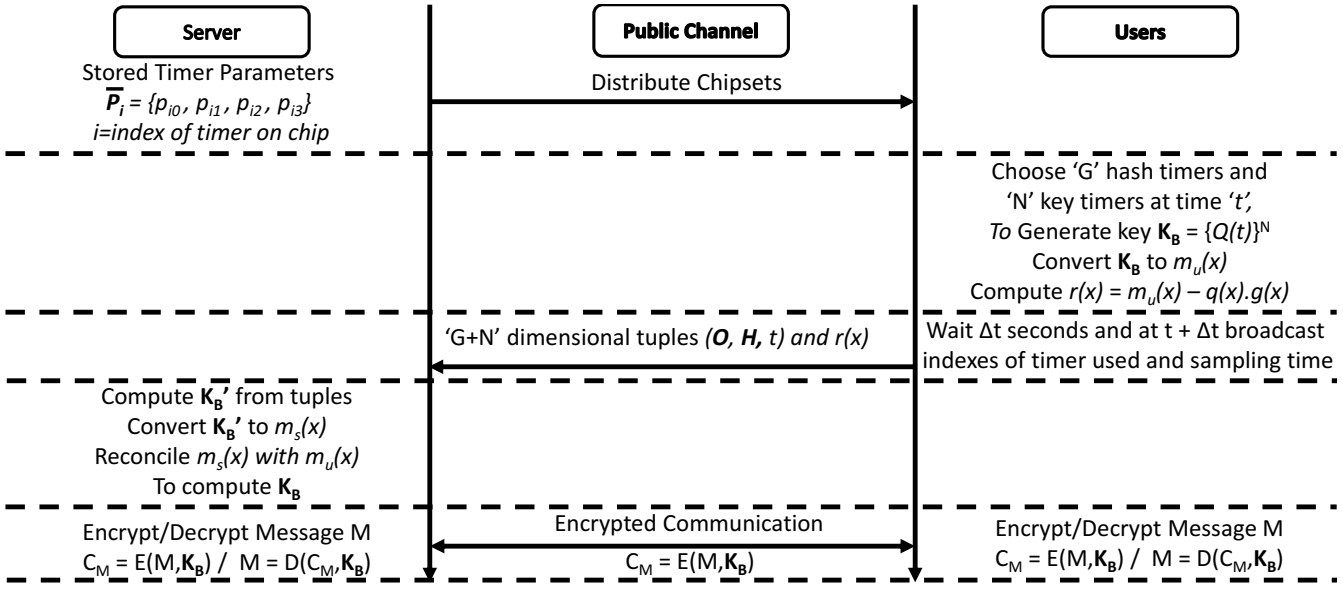

Figure 10. Modified SPoTKD protocol between a server and a user incorporating error-correction

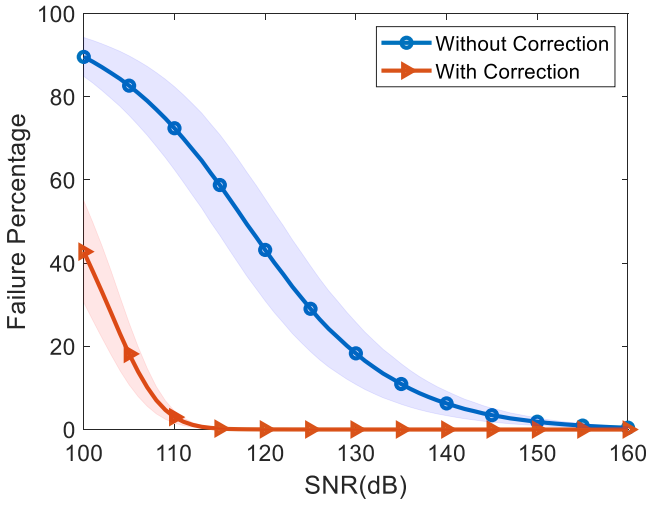

Figure 11. Performance of the SPoKTD protocol in the presence of noise when error-correction is used. A 16-bit ADC was used to measure the state of the 'key' timers. The variance across different Monte-carlo trials are highlighted by the shaded region.

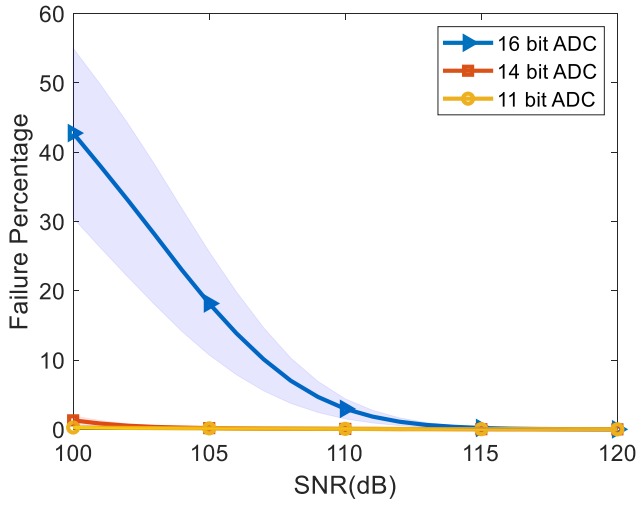

Figure 12. Performance of the SPoTKD protocol in the presence of noise when using error-correction and when the resolution b of the ADC used for measuring the state of the 'key' timer is reduced. The variance across different Monte-carlo trials are highlighted by the shaded region.

VI. Noise Robustness

In the next set of experiments, we quantified the robustness of the SPoTKD protocols in the presence of real-world operational artifacts. For instance, the timer on a physical chip could inadvertently desynchronize with the software model on the server. This could be due to fabrication mismatch, environmental variations, device degradation, and measurement noise. To emulate this effect we performed a Monte Carlo study where we added White Gaussian Noise to the timer response and then generated the keys by sampling at random time instances.

In this case, the SNR is defined as

$$SNR = \frac{P_{Signal}}{P_{Noise}}$$

where P_{Signal} is square of the signal output measured from the timer and P_{Noise} is the signal variance. This 'measured' key was compared against the 'gold' key generated from the software model in the server i.e. without any noise. Every instance where the keys do not match perfectly is counted as a failure. Figure 9 shows the failure percentage, calculated as the average number of failure instances over all the instances of simulation, at each noise level. As expected, the failure percentage reduces with an increase in SNR.

Better noise robustness could be achieved by using low-resolution ADC for the key timers, as shown in Figure 9. However, as we have shown in the previous section this could lead to more information gained by a 'knowledgeable' attacker to predict the key. In order to mitigate this threat, the server can recommend the user to opt for an increase in the wait-period Δt and achieve the same level of uncertainty even for low-resolution ADC, as illustrated in Figure 7. Thereby, a tradeoff exists between the level of security and the waiting period, and the preference for one or the other depends on the target application.

VII. Error Correcting SPoTKD

In the previous section, we have discussed how the protocol's robustness to noise could be increased by either trading off security or waiting period. In this section, we will discuss a new protocol shown in Figure 10 in which noise robustness can be improved without compromising neither security nor waiting time by using standard error-correcting codes which are generally used in digital communication. For our purpose, we will use cyclic-redundancy-check (CRC) for error correction [34], even though other error-correcting codes could also be used.

The string of key-bits are represented as the coefficients of a message polynomial, $m(x)$, over a Galois field (GF2) and to find the CRC, the message polynomial is multiplied by x^n and then the remainder $r(x)$ is found by dividing with an n-degree generator polynomial $g(x)$. The coefficients of the remainder polynomial are the bits of the CRC. This can be expressed as

$$m_u(x).x^n = q(x).g(x) + r(x) \tag{13}$$

where $q(x)$ is the quotient. Typically, $m_u(x).x^n - r(x)$ and $g(x)$ is sent over the communication channel. However, in this protocol we are sending $r(x)$ i.e. only the CRC bits together with the tuples $(\mathbf{O}, \mathbf{H}, t))$ over an insecure channel as illustrated in Figure 10, and $g(x)$ is assumed to be predetermined and a public knowledge. This is because we do not want to share the message $m_u(x)$ which is the key itself. The server generates the $m_s(x)$ using the tuples $(\mathbf{O}, \mathbf{H}, t))$ information and the software model. Then, together with $r(x)$ and $g(x)$ the sever can reconcile $m_s(x)$ with $m_u(x)$ up to a certain hamming distance. Thereby, tolerating erroneous key-bits measured by the user due to noise.

From the security point of view, the attacker now has more information about the key as the remainder $r(x)$ is broadcast along with the $(\mathbf{O}, \mathbf{H}, t))$ tuples. For example, let $m(x)$ be the representation of a 256-bit key. Then the number of possible keys = 2^{256}. We assume that the attacker has an identical chip

himself. Let $g(x)$ be a 28-degree polynomial, then with the knowledge of $r(x)$ the number of possible keys is reduced to $2^{256\ \ 28} = 2^{228}$. Therefore, the search complexity for an attacker decreases proportionally to the degree of generator polynomial used i.e. number of CRC bits.

In order to counteract this effect, the length of the key can be increased by an amount equal to the degree of $g(x)$. This would mean more timers are needed to be used for an effective key length equal to the number of timers used minus the degree of $g(x)$. In the example described above, the number of timers required for a 256-bit effective key length would be 284.

According to Philip Koopman's table of CRC generator polynomial [35], for a $g(x)$ of 28 degrees and data-word length less than 483 bit, the least hamming distance that can be corrected is 8. Therefore, we can allow up to 8 mismatches for the 284-bit key, which has an effective key length of 256-bits, and then compare the noise robustness to the 256-bit key. This is illustrated in Figure 11 which shows significant noise robustness improvement. This is achieved without sacrificing any complexity and does not come at the cost of a longer waiting period. Robustness can be further improved by using lower resolution ADC for key-generation as shown in Figure 12 if the user opts for more accuracy and is compliant with a longer waiting period.

VIII. Discussions and Conclusions

In this paper, we introduced a novel key distribution framework, SPotKD, based on specific security features of the previously reported self-powered time-keeping devices. We described the key exchange protocol and also analyzed it both from a security and noise robustness point of view. Our protocol is not only secure against most kinds of attacks but also proved to be secure in the advent of a fully functional quantum computer in the future. We have also evaluated the performance of our protocol against some state-of-the-art key distribution schemes.

Several challenges exist in implementing the proposed key distribution system from a practical point of view. At the core of the system is the self-powered timer technology which have been successfully demonstrated in our prior work [18], [27]. However, designing the peripheral circuitry that can realize the key generation protocol on-chip is yet to be accomplished. A complete system-on-chip (SoC) should consist of an array of these timers and a combinational logic circuit that will allow the user to arbitrarily choose any set of timers for key generation. In addition, the circuit for destroying the timer's information should also be integrated into the chipsets. Furthermore, the design of the destruction circuit should be done in such a manner that the timers' temporal response becomes desynchronized even before the user can access the output of the chipsets and remain desynchronized for a significant period after read-out. Only then, the timers can be considered to be a one-time read device. Addressing this challenge would be a part of future research.

Another limitation arises in scaling the framework due to the limited number of chips that can be distributed while maintaining security against quantum attacks as discussed in claim 2. However, the limit on the number of chipsets can be increased by using more hash timers during key generation. It should also be noted that due to real-world artifact noise, increasing the number of hash timers may lead to high failure rates during key exchange. The protocol will remain secure albeit slight increase in the probability of a key exchange failure and a trade-off exists between the security and the reliability of the SPoTKD protocol. In this regard, incorporating error correcting mechanisms in the SPoTKD protocol will help to address these limitations.

One other limitation to consider is that the underlying assumption for SPoTKD dictates that the server has ample resources to securely store the timer parameters and to secure access control. With respect to secure storage, the server can adopt traditional, high-end and computationally intensive symmetric key encryption approaches. However, the protocol in its current state will not remain secure if the server becomes compromised and attacker gain access to the timer initialization parameters using phishing techniques or by compromising the access control protocols (similar to the attack models demonstrated for trusted program modules [36]). This vulnerability can be overcome by adopting a distributed server (Decentralized Cloud Storage) approach. The security of these types of storage system is well established [37] where AES-256 is used to encrypt the data and then each data is split and stored across a distributed network. Another solution that we are currently investigating, is storing the timer initialization parameters in a semi-persistent storage (memory whose content is destroyed after a pre-determined time). This attribute will prevent against the "record now decode later" attacks where the attacker logs the encrypted data with the hope that a powerful computer will be available to successfully decrypt the data, or the server storage will be compromised at a much later time.

Our future work would focus on prototyping a self-powered timer system-on-chip with all the basic hardware security primitives. We will then validate the SPoTKD protocol under real-world conditions and over different distribution channels. This will open the possibility of applying SPoTKD in areas such as quantum secure blockchains (based on symmetric-key) and electronic voting.

Acknowledgments

The authors would like to thank Dr. Kenji Aono, Dr. Darshit Mehta and Dr. Sri Harsha Kondapalli at the Electrical and Systems Engineering department, Washington University in St. Louis, for their valuable assistance in running experiments and MATLAB© simulations.

References

[1] K. Aono, N. Lajnef, F. Faridazar, and S. Chakrabartty, "Infrastructural health monitoring using self-powered internet-of-things," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2058–2061, 2016.

[2] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," *IEEE Access*, vol. 5, pp. 26521–26544, 2017.

[3] U. 42, "2020 unit 42 iot threat report." https://start.paloaltonetworks.com/unit-42-iot-threat-report, 2020.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, p. 120–126, Feb. 1978.

[5] N. A. of Sciences, Engineering, and Medicine, "4 quantum computing's implications for cryptography," *Quantum Computing: Progress and Prospects*, 2019.

[6] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *In Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, vol. 175, p. 8, New York, 1984.

[7] A. K. Ekert, "Quantum cryptography based on bell's theorem," *Phys. Rev. Lett.*, vol. 67, pp. 661–663, Aug 1991.

[8] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Reviews of Modern Physics*, vol. 74, p. 145–195, Mar 2002.

[9] C. Portmann and R. Renner, "Cryptographic security of quantum key distribution," *arXiv preprint arXiv:1409.3525*, 2014.

[10] R. Ursin, F. Tiefenbacher, T. Schmitt-Manderbach, H. Weier, T. Scheidl, M. Lindenthal, B. Blauensteiner, T. Jennewein, J. Perdigues, P. Trojek, and et al., "Entanglement-based quantum communication over 144km," *Nature Physics*, vol. 3, p. 481–486, Jun 2007.

[11] A. R. Dixon, Z. L. Yuan, J. F. Dynes, A. W. Sharpe, and A. J. Shields, "Gigahertz decoy quantum key distribution with 1 mbit/s secure key rate," *Optics Express*, vol. 16, p. 18790, Oct 2008.

[12] B. Korzh, C. C. W. Lim, R. Houlmann, N. Gisin, M. J. Li, D. Nolan, B. Sanguinetti, R. Thew, and H. Zbinden, "Provably secure and practical quantum key distribution over 307km of optical fibre," *Nature Photonics*, vol. 9, p. 163–168, Feb 2015.

[13] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai, G.-B. Li, Q.-M. Lu, Y.-H. Gong, Y. Xu, S.-L. Li, F.-Z. Li, Y.-Y. Yin, Z.-Q. Jiang, M. Li, J.-J. Jia, G. Ren, D. He, Y.-L. Zhou, X.-X. Zhang, N. Wang, X. Chang, Z.-C. Zhu, N.-L. Liu, Y.-A. Chen, C.-Y. Lu, R. Shu, C.-Z. Peng, J.-Y. Wang, and J.-W. Pan, "Satellite-based entanglement distribution over 1200 kilometers," *Science*, vol. 356, no. 6343, pp. 1140–1144, 2017.

[14] N. Jain, C. Wittmann, L. Lydersen, C. Wiechers, D. Elser, C. Marquardt, V. Makarov, and G. Leuchs, "Device calibration impacts security of quantum key distribution," *Physical review letters*, vol. 107, p. 110501, 09 2011.

[15] G. Brassard, N. Lütkenhaus, T. Mor, and B. Sanders, "Limitations on practical quantum cryptography," *Physical review letters*, vol. 85, pp. 1330–3, 09 2000.

[16] R. Courtland, "Intel now packs 100 million transistors in each square millimeter." https://spectrum.ieee.org/nanoclast/semiconductors/processors/intel-now-packs-100-million-transistors-in-each-square-millimeter, 2017.

[17] G. Halfacree, "Onchip unveils itsy-chipsy ultra-low-cost ic fabrication platform." https://abopen.com/news/onchip-unveils-itsy-chipsy-ultra-low-cost-ic-fabrication-platform/, 2017.

[18] L. Zhou and S. Chakrabartty, "Self-powered timekeeping and synchronization using fowler-nordheim tunneling-based floating-gate integrators," *IEEE Transactions on Electron Devices*, vol. PP, pp. 1–7, 01 2017.

[19] R. Alléaume, C. Branciard, J. Bouda, T. Debuisschert, M. Dianati, N. Gisin, M. Godfrey, P. Grangier, T. Länger, N. Lütkenhaus, *et al.*, "Using quantum key distribution for cryptographic purposes: a survey," *Theoretical Computer Science*, vol. 560, pp. 62–81, 2014.

[20] G. Murphy, A. Keeshan, R. Agarwal, and E. Popovici, "Hardware-software implementation of public-key cryptography for wireless sensor networks," 2006.

[21] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," tech. rep., USA, 1979.

[22] A. Di Falco, V. Mazzone, A. Cruz, and A. Fratalocchi, "Perfect secrecy cryptography via mixing of chaotic waves in irreversible time-varying silicon chips," *Nature Communications*, vol. 10, 12 2019.

[23] L. Keuninckx, M. Soriano, I. Fischer, C. Mirasso, R. Nguimdo, and G. Van der Sande, "Encryption key distribution via chaos synchronization," *Scientific Reports*, vol. 7, p. 43428, 02 2017.

[24] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *International Workshop on Information Hiding*, pp. 206–220, Springer, 2009.

[25] B. Danev, H. Luecken, S. Capkun, and K. El Defrawy, "Attacks on physical-layer identification," in *Proceedings of the third ACM conference on Wireless network security*, pp. 89–98, 2010.

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

[27] L. Zhou, S. H. Kondapalli, K. Aono, and S. Chakrabartty, "Desynchronization of self-powered fn tunneling timers for trust verification of iot supply chain," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6537–6547, 2019.

[28] M. H. Afifi, L. Zhou, S. Chakrabartty, and J. Ren, "Dynamic authentication protocol using self-powered timers for passive internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2927–2935, 2018.

[29] D. Giry, "Bluekrypt:cryptographic key length recommendation." https://www.keylength.com/en/4/, 2020.

[30] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray, and S. Vo, "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," tech. rep., Gaithersburg, MD, USA, 2010.

[31] H. Wire, "Fugaku retains title as world's fastest supercomputer." https://www.hpcwire.com/off-the-wire/fugaku-retains-title-as-worlds-fastest-supercomputer/, November 17, 2020.

[32] G. A. V. R. C. Rao, P. V. Lakshmi, and N. R. Shankar, "Article: Rsa public key cryptosystem using modular multiplication," *International Journal of Computer Applications*, vol. 80, pp. 38–42, October 2013. Full text available.

[33] T. Calver, M. Grimaila, and J. Humphries, "An empirical analysis of the cascade error reconciliation protocol for quantum key distribution," in *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW '11, (New York, NY, USA), Association for Computing Machinery, 2011.

[34] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[35] P. Koopman, "Best crc polynomials." https://users.ece.cmu.edu/ koopman/crc/, 2015.

[36] D. Group, "From stolen laptop to inside the company network." https://dolosgroup.io/blog/2021/7/9/from-stolen-laptop-to-inside-the-company-network, July 28, 2021.

[37] Cryptopedia, "An overview of decentralized cloud storage services." https://www.gemini.com/cryptopedia/crypto-cloud-storage-decentralized-cloud-storage-providers, December 21, 2021.

Mustafizur Rahman received his B.S. degree in Electrical Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2017. He is presently a doctoral candidate in the Electrical and Systems Engineering Department at Washington University in St. Louis.

His current research interests include analog circuits, hardware security and memory design.

Liang Zhou received the B.S. degree in physics from Tsinghua University, Beijing, China, in 2010, and the Ph.D. degree in computer engineering from Washington University in St. Louis, St. Louis, MO, USA, in 2018.

He is a Design Engineer with Analog Devices Inc., Grass Valley, CA, USA. His current research interests include low-power sensing systems, analog and mixed-signal circuits, and hardware security.

Dr. Zhou was a recipient of the Best Paper Award and the Honorary Mention for Best Paper Award presented by the ISCAS in 2013 and 2015, respectively.

Shantanu Chakrabartty (S'99-M'04-SM'09) received his B.Tech degree from Indian Institute of Technology, Delhi in 1996, M.S and Ph.D degrees in Electrical Engineering from Johns Hopkins University, Baltimore, MD, USA in 2002 and 2004 respectively.

From 1996 to 1999, he was with Qualcomm Inc., San Diego, CA, USA. In 2002, he was a visiting researcher at The University of Tokyo, Tokyo, Japan. From 2004 to 2015, he was an Associate Professor with the Department of Electrical and Computer Engineering, Michigan State University (MSU), MI, USA. He is currently a Clifford Murphy Professor in the Department of Electrical and Systems Engineering at Washington University in St. Louis, MO, USA. Dr. Chakrabartty's work covers different aspects of analog computing, and his current research interests include self-powered sensors and neuromorphic and hybrid circuits and systems.

Dr. Chakrabartty was a Catalyst foundation fellow from 1999-2004 and is a recipient of National Science Foundation's CAREER award, University Teacher-Scholar Award from MSU and the 2012 Technology of the Year Award from MSU Technologies. Dr. Chakrabartty is a fellow of the American Insitute of Medical and Biological Engineering (AIMBE), a senior member of the IEEE and has previously served as the associate editor for IEEE Transactions of Biomedical Circuits and Systems and an associate editor for Frontiers of Neuromorphic Engineering journal.