PUF-FSM: A Controlled Strong PUF

Yansong Gao, *Student Member, IEEE*, Hua Ma, Said F. Al-Sarawi, *Member, IEEE*, Derek Abbott, *Fellow, IEEE*, and Damith C. Ranasinghe, *Member, IEEE*

***Abstract*—Existing strong controlled physical unclonable function (PUF) designs are built to resist modeling attacks and they deal with noisy PUF responses by exploiting error correction logic. These designs are burdened by the costs of the error correction logic and information shown to leak through the associated helper data for assisting error corrections; leaving the design vulnerable to fault attacks or reliability-based attacks. We present a hybrid PUF–finite state machine (PUF-FSM) construction to realize a controlled strong PUF. The PUF-FSM design removes the need for error correction logic and related computation, storage of the helper data and loading it on-chip by *only* employing error-free responses judiciously determined on demand in the absence of the underlying PUF—an Arbiter PUF—with a large challenge response pair space. The PUF-FSM demonstrates improved security, especially to reliability-based attacks and is able to support a range of applications from authentication to more advanced cryptographic applications built upon shared keys. We experimentally validate the practicability of the PUF-FSM.**

***Index Terms*—Arbiter physical unclonable function (APUF), error-free responses, fault attacks, modeling attacks, physical uncloanble function (PUF), statistical model.**

I. Introduction

The physical unclonable function (PUF), a hardware security primitive, exploits manufacturing variations to extract secrecy on demand [1], [2]. Since the advent of the silicon Arbiter PUF (APUF) in 2002 [3], the PUF community has been pursuing so-called strong PUFs that not only have a very large challenge response pair (CRP) space but are also resilient to modeling attacks. The instance-specific CRP behavior of strong PUFs is naturally appealing for lightweight authentication protocols [2], [4], [5]. Beyond authentication, strong PUFs are also employed for key generation and more advanced cryptographic protocols, e.g., key exchange and oblivious transfer [6], [7]. However, a practical lightweight strong PUF realization compatible with current CMOS technology remains a challenging proposition in the face of modeling attacks responsible for breaking previously deemed strong PUFs including the XOR-APUF, feedforward APUF, lightweight secure PUF [8], and even the slender PUF [9], [10].

Manuscript received March 3, 2017; revised May 31, 2017; accepted July 7, 2017. Date of publication August 15, 2017; date of current version April 19, 2018. This work was supported in part by the Australian Research Council Discovery Project under Grant DP140103448, and in part by the China Scholarship Council under Grant 201306070017. This paper was recommended by Associate Editor S. Bhunia. *(Corresponding author: Yansong Gao.)*

Y. Gao is with the School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China, also with the School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia, and also with the Auto-ID Laboratory, School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia (e-mail: yansong.gao@njust.edu.cn).

H. Ma and D. C. Ranasinghe are with the Auto-ID Laboratory, School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia (e-mail: mary.ma@adelaide.edu.au; damith.ranasinghe@adelaide.edu.au).

S. F. Al-Sarawi and D. Abbott are with the School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia (e-mail: said.alsarawi@adelaide.edu.au; derek.abbott@adelaide.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2017.2740297

Yu *et al.* [11] recently presented a practical strong PUF by placing an upper bound on the available number of CRPs to an adversary. To be precise, gaining new CRPs has to be explicitly authorized by the trusted entity—this concept of limiting access to CRPs is akin to controlled PUFs (C-PUFs) [12], detailed in Section II-B. Yu *et al.* [11] further introduced a PUF device side nonce to prevent fault attacks or reliability-based attacks [9], [10].

We continue the pursuit of a practical and lightweight controlled strong PUF dubbed the PUF-finite state machine (FSM). Through the ability to use challenges capable of generating error free responses, the PUF-FSM is able to generate key bits without error correcting post-processes that have hitherto been always required in PUF-based key generators [13]. Further, the ability to remove the information leakage, possible with the use of error correcting codes recently exploited in [9], [10], and [14], essentially removes an attack vector from a PUF-FSM key generator. For authentication, the PUF-FSM avoids one major limitation of [11] in terms of available secure authentication rounds. Contributions of this paper are as follows.

1) We propose a design for a controlled strong PUF—PUF-FSM—with improved lightweight characteristics by employing a large number of available error-free responses determined in the absence of the underlying strong PUF; an APUF in this paper. The PUF-FSM, to the best of our knowledge, is the first C-PUF without using error correction code (ECC) logic along with the associated helper data and with an explicit design consideration to counter reliability-based attacks.
2) We demonstrate that the PUF-FSM has significantly improved security through its resilience to various plausible modeling attacks including reliability-based attacks in [14]. We validate the practicability of the PUF-FSM through experimental results, compare it with other related works, and briefly discuss its applicability to a range of security applications.

Section II introduces related work. Section III details the PUF-FSM design and analyses its security. Experimental validation of the PUF-FSM is performed in Section IV while applications are presented in Section V.

II. Related Work

A. APUF Model for Error-Free Response Generation

The APUF consists of k stages of two 2-input multiplexers or any other units forming two signal paths [1]. To generate a response bit, a signal is applied to the first stage input, while the challenge $\mathbf{C}$ determines the signal path to the next stage. The input signal will race through each multiplexer path (top and bottom paths) in parallel with each other. At the end of the APUF architecture, an Arbiter, e.g., a latch, determines whether the top or bottom signal arrives first and hence results in a logic "0" or "1" accordingly. Hence, it is the time delay difference, t_{dif}, of an APUF challenge that determines its response.

It has been shown that an APUF can be modeled, where the APUF response given an unseen challenge can be predicted with high accuracy using a learned model—usually with modeling attacks employing machine learning techniques [8]. Thus, the t_{dif} can

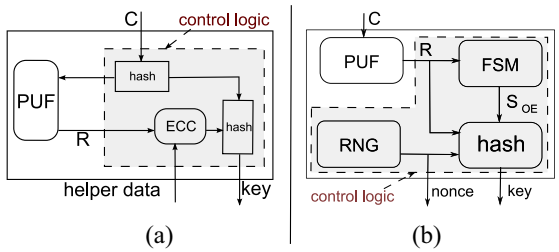

Fig. 1. (a) C-PUF construction [12], [17]. (b) General structure of the PUF-FSM. Only the correct sequential challenges $\mathbf{C}_{\text{set}}$ can activate the enable signal S_{OE}. If the enable signal S_{OE} is disabled, the hash output is meaningless as it is a result of random values. Otherwise, a key is generated based on *part* of the response $\mathbf{R}$, $\mathbf{R}_{\text{secret}}$, and the *nonce*, where the key = HASH($\mathbf{R}_{\text{secret}}$, nonce).

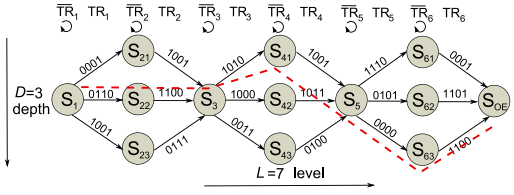

Fig. 2. FSM example with seven levels ($L = 7$) and three depths ($D = 3$).

eventually be predicted based on the learned APUF model without physical measurements. Recent works [15], [16] show that t_{dif} comprises of two important pieces of information: 1) the sgn(t_{dif}) determines the binary response and 2) the magnitude of t_{dif} indicates the reliability of this response. If the t_{dif} is far from zero, then this gives full confidence that such a challenge can reproduce the response without an error. In PUF-FSM, we show how to exploit the bit level reliability information from such a model to build a secure, controlled strong PUF.

B. Controlled PUF

The C-PUF [12], [17] proposed by Gassend *et al.* is a strong PUF construction. It combines an underlying PUF with control logic limiting the ways in which the PUF can be evaluated. In practice, the C-PUF is built so that the underlying PUF and control logic play complementary roles. As illustrated in Fig. 1(a), the PUF prevents invasive attacks on the control logic, whilst the control logic protects the PUF from protocol level attacks. For example, the control logic is enclosed by metal wire tracks running above and below the circuitry. These tracks introduce path delays that the APUF uses to determine its response—so if the tracks are broken while physically probing the control circuitry, the digital secret is altered or destroyed [12]. The challenge and response in a C-PUF are pre- and post-processed, e.g., hashed, respectively, and thus the control logic can halt adaptive evaluations of PUFs without permission from a trusted entity [11]. Previous studies [12], [17] require ECC logic and the associated helper data to stabilize the noisy PUF responses before hashing. However, the ECC logic is usually expensive, especially for low-end devices. Most importantly, employment of helper data exposes the C-PUF to modeling attacks exploiting noise side-channel information [10], [14], [18].

C. FSM-Based Locking Mechanisms

Recent FSM-based locking methods together with PUFs have been used in the literature for applications, such as IC active metering, protecting or locking intellectual property (IP) [19]–[21], and preventing netlist reverse-engineering [22]. For instance, HARPOON [22] is a gate-level obfuscation-based design that provides security at multiple stages of the IC life cycle during fabrication, test and deployment. The work in [21] enables an field-programmable gate array (FPGA) user to pay a license fee for the specific IP and FPGA employed and, thus, reduce costs incurred by users while protecting the revenue of the IP core developer and FPGA vendor. In these studies, an FSM is used as an obfuscation technique, where stability of PUF responses need to be specifically considered and the aim is not to realize a PUF-based security primitive.

In contrast, we address instability of PUF responses by winnowing the challenges that generate unstable responses and combine this approach with an FSM and a random number generator (RNG)-based response obfuscation method to: 1) hide the direct relationship between challenges and responses; 2) eliminate the attacks that exploit reliability information from PUF response bits obtained from repeated evaluations; and 3) still realize an exponential challenge-response space to build a strong C-PUF—a cryptographic primitive. Using a PUF-FSM architecture to realize a strong PUF-based security primitive creates the ability to build multiple security services, as highlighted in Section V.

III. PUF-FSM: Design and Security Analysis

A. PUF-FSM Structure

A generalized construction of a PUF-FSM shown in Fig. 1(b). It consists of an underlying PUF, an FSM, a hash function and an RNG block. Similar to prior work [11], the direct PUF responses can only be evaluated by a trusted entity (e.g., the server) in a secure environment during the PUF provisioning phase to build APUF statistical model(s), and the direct access is destroyed afterwards, e.g., through fusing a wire.

During deployment, a set of n sequential challenges, $\mathbf{C}_{\text{set}}$, is issued by the trusted entity and applied to the PUF-FSM, corresponding error-free responses $\mathbf{R}$ with length n are produced internally. The $\mathbf{R}$ is sequentially fed into the FSM to direct the transitions of the FSM states, where the FSM is reset to S_1 before the operation. Only a series of correct transition conditions $\mathtt{TR}$—a specific response substrings enabling a state transition to the next state—is able to guarantee the FSM traversing through to the S_{OE} state to activate the key output. Thus, only the server who possesses the APUF model is capable of issuing a correct $\mathbf{C}_{\text{set}}$ to activate the S_{OE} to generate a meaningful output as a key. The key is HASH($\mathbf{R}_{\text{secret}}$, nonce), where $\mathbf{R}_{\text{secret}}$ is a substring of $\mathbf{R}$; formation of $\mathbf{R}_{\text{secret}}$ will be described soon. The RNG prepares two random strings, the output presents the first random string to replace the $\mathbf{R}_{\text{secret}}$ in the formation of the key when the S_{OE} is disabled, the second random string forms the nonce.

An exemplary FSM construction is depicted in Fig. 2. At the beginning of the PUF-FSM operation, the FSM resets to its initial state S_1. Let us assume that the $\mathtt{TR}_1$ is 0110, then $S_1 \xrightarrow{0110} S_{22}$. Similarly if the $\mathtt{TR}_1 = 0001$, then $S_1 \xrightarrow{0001} S_{21}$. If $\mathtt{TR}_1 \notin \{0001, 0110, 1001\}$, or in other words the input is in $\overline{\mathtt{TR}}_1$, then $S_1 \xrightarrow{\overline{\mathtt{TR}}_1} S_1$—a self transition, where the FSM remains at its current state. In this example, for odd states S_1, S_3, S_5, there are D edges or transitions conditioned on D inputs to lead to one of the following D states while transitions from other states are conditioned on a single input.

While other FSM structures can be envisioned, the FSM in this case study has L—always an odd number—of state layers (levels), where each even internal layer has D parallel states. At least $L - 1$ transitions are required to reach the S_{OE} state. Both $\mathtt{TR}_l$ and $\overline{\mathtt{TR}}_l$ are 4-bit in our example, therefore the *maximum* number of transitions that can be employed is $n/4$, where we assume that n is always a multiple of 4 for convenience. Given that we need at least $L - 1$

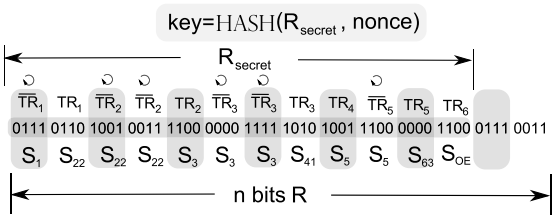

Fig. 3. Part of the n-bit $\mathbf{R}$, $\mathbf{R}_{\text{secret}}$, is hashed to generate the key. All the remaining bits after reaching the S_{OE} state are not contributing to the key (here we use the FSM example in Fig. 2 and the set of transitions followed are marked by the red dotted line).

transitions, i.e., $\texttt{TR}_i$ for $i = \{1, \ldots, L-1\}$, the maximum number of self-transitions $\overline{n}_{\max} = (n/4) - (L-1)$. In practice, S_{OE} can be activated in a manner that employs $L-1$ of $\texttt{TR}_l$ and $\overline{n}$ of $\overline{\texttt{TR}}_l$ inputs subjected to $\overline{n} \leq \overline{n}_{\max}$. A meaningful key will be generated only after all n bits in $\mathbf{R}$ are fed into the FSM—or after n clock cycles—*and* the S_{OE} state is reached. The key is a hash of a part of the response $\mathbf{R}$, specifically, the input bits defined by the sequence of $\texttt{TR}_l$ and $\overline{\texttt{TR}}_l$ employed to reach the S_{OE} state. Illustration of the key formation is shown in Fig. 3, while the traversed path is illustrated in Fig. 2 using the dotted red line. Once S_{OE} state is reached, the rest of the input sequence is ignored and will not be used to form the $\mathbf{R}_{\text{secret}}$ hashed to generate the key. It is worth stressing here that the response bits beyond $\mathbf{R}_{\text{secret}}$ are still fed into the FSM as redundant bits to hide the length of $\mathbf{R}_{\text{secret}} = (L-1) + \overline{n}$, where $\overline{n}$ is determined, and therefore only known, by the server.

B. Security Analyses

1) Adversary Model: We adopt the adversary model used for the C-PUF [12], [17]. Hence: 1) physical attacks on the control logic or an attempt to probe the internal PUF will alter or even destroy the PUF itself and 2) an adversary is capable of eavesdropping on the communication channel and arbitrarily applying challenges to the PUF-FSM interface to observe the output. Furthermore, we assume the nonce is visible and the netlist of the PUF-FSM design is publicly known. Then, the objective of the adversary is to obtain useful information to learn an accurate model of the underlying APUF.

2) Brute-Force Attacks: The probability of finding a meaningful key by guessing a correct $\mathbf{C}_{\text{set}}$ without the assistance from the trusted entity is

$$P = \left(\frac{D}{2^{n_{\text{TR}}}}\right)^{\frac{L-1}{2}} \times \left(\frac{1}{2^{n_{\text{TR}}}}\right)^{\frac{L-1}{2}} \tag{1}$$

where the n_{TR} is the length of a $\texttt{TR}_l$. In the example in Fig. 2, the $n_{\text{TR}} = 4$. For each odd layer (e.g., S_1 and S_3), the probability of guessing n_{TR} challenges producing a correct transition edge is $D/2^{n_{\text{TR}}}$, while the probability of guessing n_{TR} challenges producing a correct transition edge given an even layer is $1/2^{n_{\text{TR}}}$. As an example, setting $L = 41$ and $D = 3$ implies that $P \approx 1/2^{128}$, thus, making a successful attack infeasible.

3) Modeling Attacks: In PUF-FSM, arbitrary CRP collection is disabled from any party except the trusted entity during the secure enrollment phase. Subsequently, the generation of the response is controlled by the server through the FSM and only the hash of the response and a nonce are exposed. Thus, modeling attacks requiring direct challenge-response pair information [8] are prevented since the responses generated from a PUF-FSM no longer bear a relationship to the received challenge.

Unlike previous modeling attacks, recent reliability-based modeling attacks [10] do not require the knowledge of the response for a given challenge but only the binary reliability of the generated response from the challenge.

Now, we examine a method, similar to fault-injection attacks, for discovering challenges that produce unreliable responses by observing the PUF-FSM output when a device nonce is *not* used. We assume that an eavesdropping adversary has a potential way to determine the reliability of a response to a challenge through exhaustive search under the condition that *a priori* challenge set $\mathbf{C}_{\text{set}}$ has been eavesdropped. We also assume that the adversary has full access to the PUF-FSM's challenge-response interface. The adversary chooses an unused challenge $\mathbf{c}_x \notin \mathbf{C}_{\text{set}}$ to replace one challenge $\mathbf{c}_i$ in the eavesdropped challenge set $\mathbf{C}_{\text{set}}$ to observe the output of the PUF-FSM. Then through repeated evaluations, the adversary can determine if $\mathbf{c}_x$ generates unreliable responses by simply observing any alterations in the hashed output of the PUF-FSM. Hence, through exhaustive searching, other unreliable challenges can be determined to mount a reliability-based modeling attack.

However, employing a nonce, as in [11], prevents an adversary from observing variations in the hashed output using repeated queries to the PUF-FSM with adaptive challenges to gather reliability related information based on eavesdropped $\mathbf{C}_{\text{set}}$. This is because the nonce is refreshed and hashed with the APUF response for each evaluation. Thus, the discovery of unreliable challenges through adaptive and repeated queries to mount a reliability-based modeling attack [9], [10] is prevented.

4) Optical Emission Attacks: Without proper protections, similar to nonvolatile memories, APUFs are vulnerable to optical emission-based attacks [23]. However, these attacks usually require significant skills and specialized equipment. Nonetheless, there are several countermeasures against optical emission attacks [24]. Interconnect meshes can be used to prevent optical emission attacks performed from a chip's frontside; for backside optical emission attacks, through-silicon-via technologies can be exploited as a countermeasure. Additionally, optical interaction is a low-cost alternative to protecting ICs against backside optical emission attacks [24].

5) Timing and Power Attacks: Although, power and timing side-channel attacks together with machine learning methods have been demonstrated in attacks to break APUFs [14], [25], these attacks appear to be inapplicable in practice. For example, the required accurate timing measurements may not be available on-chip [14]. Power analysis attacks require low noise levels when repeatedly measuring power traces. Therefore, in practice, an efficient countermeasure is to inject noise into the circuit to prevent accurate power trace measurements. For example, the RNG utilized in the PUF-FSM can be operated in parallel when the responses are evaluated from the APUF without extra area overhead to inject algorithmic noise that is hard to be eliminated [14].

IV. Experimental Validation and Comparison

A. Experiment Setup and Results

We use a CRP dataset obtained from eight programmable delay line (PDL) APUFs implemented in eight different FPGAs. Notably, the PDL APUF has the same topology of the multiplexer-based APUFs [16]. Each APUF is fed with 64 000 challenges, therefore, 64 000 CRPs are collected [26], [27]. Each CRP is evaluated 128 times at the same operating condition. Three temperature settings (5 °C, 35 °C, and 65 °C) and three voltage settings (0.95 V, 1.00 V, and 1.05 V) under a given temperature setting are tested.

We treat (35 °C, 1.00 V) as the nominal condition. We use 10 000 CRPs that are evaluated under the nominal condition to learn an APUF model. It takes less than 15 s to enroll each APUF by using

MATLAB 2012b software running on Intel i7-3770CPU@3.4 GHz CPU with 16 GB RAM.

B. Error-Free Response Selection Results

The APUF model has a prediction rate of 92.41%. This lower rate attributes to the usage of PDL-based APUFs on FPGAs, that tend to be more noisy, rather than multiplexer-based APUFs in an application-specified integrated circuit (ASIC) design. We found that the maximum PUF response error rate is 16.68%, which is almost four times larger than the 4.3% error rate of the ASIC APUF in [11].

By implementing reliable response selection, we are able to select more than 1.8×10^{17} error-free responses from a 64-stage APUF even in the presence of response error rates up to 16.68%. Such an extremely large number of available error-free responses is adequate for almost all practical applications. In addition, the error-free selection based on APUF model does not deteriorate other PUF performance metrics, such as the randomness. We refer interested readers to [16] for details.

C. Comparisons

We focus on comparing the PUF-FSM with the original C-PUF [12], [17] and the recent lockdown PUF [11]. As highlighted in [11] and [28], the area overhead is the most significant concern while power and delay overheads have minor impact as the security services, such as authentication and key generation built upon a C-PUF are not continuously employed operations. Thus, we concentrate on assessing and comparing the area overhead.

In Table I, all three works employ APUFs. A 64-stage APUF implemented in an ASIC design requires 260 gate equivalent (GE) (where a GE is equivalent to the area required by a two-input NAND gate [11]). The lockdown PUF [11] uses four APUFs, while the other approaches are built on a single APUF. We selected the SPONGENT lightweight block cipher, which costs 737 GE to produce a 128-bit output, for hashing responses [29]. The FSM is implemented by D flip-flops; each consuming five GE. Hence, we can see that the area overhead of the FSM is relatively small and negligible as also demonstrated in [20]–[22]. For example, the number of flip-flops needed is $\log_2 N_{\text{state}}$, where N_{state} is the number of states of the FSM; thus, even if 128 states are utilized, only 35 GE are needed to realize the FSM. The RNG can be implemented by using metastable PUF responses either from static random access memory PUFs [30] or APUFs [31] by exploiting available on-chip resources. Such an RNG eliminates potential reliability-based modeling attacks on the C-PUF [10], [14] as detailed in Section III-B. Nonetheless, the true random number generator (TRNG) is not restricted to PUF-based realizations, one can use other TRNG designs [32], [33].

As shown in Table I, the area overhead of the PUF-FSM is almost halved in comparison with the original C-PUF.[1] In particular, the PUF-FSM eliminates the attack vectors exposed by the requirement for error correction and helper data, such as: 1) helper data manipulation when it is stored off-chip and loaded on-chip during key generation and 2) reliability-based modeling attacks [10], [14]. Furthermore, we also remove the on-chip storage burdens imposed by on-chip helper data [18]. In comparison with the PUF lockdown technique [11] with similar overhead, the PUF-FSM eschews the limitation imposed on the number of authentication rounds. In addition, as a security primitive, the PUF-FSM can support applications beyond authentication as detailed in Section V. Further, the PUF-FSM, through the judicious challenge selection method, realizes significantly improved tolerance to response unreliability without ECC logic.

[1] Area is not reported for the C-PUF in [12] and [17]. We use the optimized ECC decoder area in [28]. We exclude the on-chip helper data storage overhead and we assume the pre- and post-hash, see Fig. 1(a), share the same hash logic, to obtain a minimum cost estimate for C-PUF.

TABLE I
COMPARING THE PUF-FSM

	C-PUF [11], [16]	Lockdown PUF [10]	PUF-FSM
ECC and helper data	✓	X	X
TRNG	X	✓	✓
Rounds of Authentication	∞	1000 (typical)	∞
Key generation and other applications	✓	X	✓
Tolerated error rate	15.0%	4.3%	16.7%
Area (GE)	2.1k	1.1k	1.1k
Reliability-based modeling attacks	✓	X	X

V. APPLICATIONS

A. Mutual Authentication

Recall that only a trusted entity has the capability of issuing a correct challenge sequence to activate the S_{OE} signal. As a consequence, only the PUF-FSM device and the trusted entity can compute $\mathbf{R}_{\text{secret}}$. When the PUF-FSM is transferred to the user, the trusted entity generates a $\mathbf{C}_{\text{set}}$ and transmits it to the user, possibly through an insecure communication channel. The user presents the $\mathbf{C}_{\text{set}}$ to the PUF-FSM and sends the PUF-FSM response—nonce and the hashed output—back to the trusted entity. The trusted entity, by virtue of the underlying APUF model, can *emulate* responses and hence *compute* HASH($\mathbf{R}_{\text{secret}}$, nonce), and compares it with the received hash value. A match authenticates the PUF-FSM. Once the PUF-FSM device is authenticated, the user applies the same $\mathbf{C}_{\text{set}}$ to the PUF-FSM to obtain a refreshed response. The user requests the refreshed response computed by the trusted entity after transmitting only the nonce to the trusted entity. The trusted entity is authenticated by the user only if the computed output received is the same as the output produced by the PUF-FSM held by the user. Thus, the PUF-FSM realizes mutual authentication.

B. Key Exchange

Consider a key exchange scenario between a user and a trusted entity. The user applies a shared $\mathbf{C}_{\text{set}}$ and sends the nonce to the trusted entity. Now only the user who holds the PUF-FSM and the trusted entity can generate the shared key. The user obtains it from the PUF-FSM, while the server *computes* it by hashing the $\mathbf{R}_{\text{secret}}$ and the nonce—notably, a key (shared key) is never directly exchanged between the parties. Such a shared key between two parties enables a wide variety of standard cryptographic protocols [12].

VI. CONCLUSION

We have presented a practical controlled strong PUF, PUF-FSM, by: 1) exploiting error-free responses determined in absence of an APUF and 2) controlling the means of evaluating the PUF by using an FSM-based control logic. As a C-PUF, PUF-FSM holds the promise of a cost-effective way to increase resistance to various attacks. We experimentally validated its practicability and compare it with other related C-PUF designs.

REFERENCES

[1] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. Design Autom. Conf. (DAC)*, San Diego, CA, USA, 2007, pp. 9–14.

[2] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "DRAM-based intrinsic physically unclonable functions for system-level security and authentication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 3, pp. 1085–1097, Mar. 2017.

[3] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. Conf. Comput. Commun. Security*, Washington, DC, USA, 2002, pp. 148–160.

[4] D. C. Ranasinghe and P. H. Cole, "Confronting security and privacy threats in modern RFID systems," in *Proc. IEEE 14th Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, USA, 2004, pp. 2058–2064.

[5] C. Zhou, K. K. Parhi, and C. H. Kim, "Secure and reliable XOR Arbiter PUF design: An experimental study based on 1 trillion challenge response pair measurements," in *Proc. 54th ACM Annu. Design Autom. Conf.*, Austin, TX, USA, 2017, Art. no. 10.

[6] U. Rührmair and M. Van Dijk, "PUFs in security protocols: Attack models and security evaluations," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, 2013, pp. 286–300.

[7] Y. Gao, H. Ma, D. Abbott, and S. F. Al-Sarawi, "PUF sensor: Exploiting PUF unreliability for secure wireless sensing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, to be published, doi: 10.1109/TCSI.2017.2695228.

[8] U. Rührmair *et al.*, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013.

[9] G. T. Becker, "The gap between promise and reality: On the insecurity of XOR Arbiter PUFs," in *Cryptographic Hardware and Embedded Systems—CHES*. Heidelberg, Germany: Springer, 2015, pp. 535–555.

[10] G. T. Becker, "On the pitfalls of using Arbiter-PUFs as building blocks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 8, pp. 1295–1307, Aug. 2015.

[11] M.-D. Yu *et al.*, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 3, pp. 146–159, Jul./Sep. 2016.

[12] B. Gassend *et al.*, "Controlled physical random functions and applications," *ACM Trans. Inf. Syst. Security*, vol. 10, no. 4, 2008, Art. no. 3.

[13] M. Hiller, "Key derivation with physical unclonable functions," Ph.D. dissertation, Inst. Security Inf. Technol., Technische Universitat Munchen, Munich, Germany, 2016.

[14] G. T. Becker and R. Kumar, "Active and passive side-channel attacks on delay based PUF designs," *IACR Cryptol. ePrint Archive*, vol. 2014, p. 287, 2014.

[15] X. Xu, W. Burleson, and D. E. Holcomb, "Using statistical models to improve the reliability of delay-based PUFs," in *Proc. Symp. VLSI*, Pittsburgh, PA, USA, 2016, pp. 547–552.

[16] Y. Gao *et al.*, "Exploiting PUF models for error free response generation," *arXiv preprint arXiv:1701.08241*, 2017.

[17] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Controlled physical random functions," in *Proc. IEEE Annu. Comput. Security Appl. Conf.*, Las Vegas, NV, USA, 2002, pp. 149–160.

[18] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 889–902, Jun. 2015.

[19] F. Koushanfar and G. Qu, "Hardware metering," in *Proc. Design Autom. Conf.*, Las Vegas, NV, USA, 2001, pp. 490–493.

[20] F. Koushanfar, "Provably secure active IC metering techniques for piracy avoidance and digital rights management," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 51–63, Feb. 2012.

[21] J. Zhang, Y. Lin, Y. Lyu, and G. Qu, "A PUF-FSM binding scheme for FPGA IP protection and pay-per-device licensing," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1137–1150, Jun. 2015.

[22] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.

[23] S. Tajik *et al.*, "Photonic side-channel analysis of arbiter PUFs," *J. Cryptol.*, vol. 30, no. 2, pp. 550–571, 2017.

[24] C. Boit *et al.*, "From IC debug to hardware security risk: The power of backside access and optical interaction," in *Proc. IEEE Int. Symp. Phys. Failure Anal. Integr. Circuits*, Singapore, 2016, pp. 365–369.

[25] U. Rührmair *et al.*, "Efficient power and timing side channels for physical unclonable functions," in *Cryptographic Hardware and Embedded Systems*. Heidelberg, Germany: Springer, 2014, pp. 476–492.

[26] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Trans. Reconfig. Technol. Syst.*, vol. 2, no. 1, 2009, Art. no. 5.

[27] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, Seattle, WA, USA, 2010, pp. 1–6.

[28] V. Van der Leest, B. Preneel, and E. Van der Sluis, "Soft decision error correction for compact memory-based PUFs using a single enrollment," in *Cryptographic Hardware and Embedded Systems*. Heidelberg, Germany: Springer, 2012, pp. 268–282.

[29] A. Bogdanov *et al.*, "SPONGENT: The design space of lightweight cryptographic hashing," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2041–2053, Oct. 2013.

[30] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[31] D. C. Ranasinghe, D. Lim, S. Devadas, D. Abbott, and P. H. Cole, "Random numbers from metastability and thermal noise," *Electron. Lett.*, vol. 41, no. 16, pp. 891–893, 2005.

[32] S. Srinivasan *et al.*, "2.4GHz 7mw all-digital PVT-variation tolerant true random number generator in 45nm CMOS," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, 2010, pp. 203–204.

[33] F. Tehranipoor, W. Yan, and J. A. Chandy, "Robust hardware true random number generators using DRAM remanence effects," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust*, McLean, VA, USA, 2016, pp. 79–84.